Efficient Generation of Random Bits From Finite State Markov Chains

Hongchao Zhou and Jehoshua Bruck, *Fellow, IEEE*

***Abstract*—The problem of random number generation from an uncorrelated random source (of unknown probability distribution) dates back to von Neumann's 1951 work. Elias (1972) generalized von Neumann's scheme and showed how to achieve optimal efficiency in unbiased random bits generation. Hence, a natural question is what if the sources are correlated? Both Elias and Samuelson proposed methods for generating unbiased random bits in the case of correlated sources (of unknown probability distribution), specifically, they considered finite Markov chains. However, their proposed methods are not efficient or have implementation difficulties. Blum (1986) devised an algorithm for efficiently generating random bits from degree-2 finite Markov chains in expected linear time, however, his beautiful method is still far from optimality on information-efficiency. In this paper, we generalize Blum's algorithm to arbitrary degree finite Markov chains and combine it with Elias's method for efficient generation of unbiased bits. As a result, we provide the first known algorithm that generates unbiased random bits from an arbitrary finite Markov chain, operates in expected linear time and achieves the information-theoretic upper bound on efficiency.**

***Index Terms*—Markov chain, Random bits generation, Random sequence.**

I. Introduction

THE problem of random number generation dates back to von Neumann [9] who considered the problem of simulating an unbiased coin by using a biased coin with unknown probability. He observed that when one focuses on a pair of coin tosses, the events HT and TH have the same probability (H is for "head" and T is for "tail"); hence, HT produces the output symbol 0 and TH produces the output symbol 1. The other two possible events, namely, HH and TT, are ignored, namely, they do not produce any output symbols. More efficient algorithms for generating random bits from a biased coin were proposed by Hoeffding and Simons [7], Elias [4], Stout and Warren [17] and Peres [12]. Elias [4] was the first to devise an optimal procedure in terms of the information efficiency, namely, the expected number of unbiased random bits generated per coin toss is asymptotically equal to the entropy of the biased coin. In addition, Knuth and Yao [8] presented a simple procedure for generating sequences with arbitrary probability distributions from an unbiased coin (the probability of H and T is $\frac{1}{2}$). Han and Hoshi [5] generalized this approach and considered the case where the given coin has an arbitrary known bias.

In this paper, we study the problem of generating random bits from an arbitrary and unknown finite Markov chain (the transition matrix is unknown). The input to our problem is a sequence of symbols that represent a random trajectory through the states of the Markov chain—given this input sequence our algorithm generates an independent unbiased binary sequence called the output sequence. This problem was first studied by Samuelson [14]. His approach was to focus on a single state (ignoring the other states) treat the transitions out of this state as the input process, hence, reducing the problem of correlated sources to the problem of a single "independent" random source; obviously, this method is not efficient. Elias [4] suggested to utilize the sequences related to all states: Producing an "independent" output sequence from the transitions out of every state and then pasting (concatenating) the collection of output sequences to generate a long output sequence. However, neither Samuelson nor Elias proved that their methods work for arbitrary Markov chains, namely, they did not prove that the transitions out of each state are independent. In fact, Blum [2] probably realized it, as he mentioned that: i) "Elias's algorithm is excellent, but certain difficulties arise in trying to use it (or the original von Neumann scheme) to generate bits in expected linear time from a Markov chain," and ii) "Elias has suggested a way to use all the symbols produced by a Markov Chain (MC). His algorithm approaches the maximum possible efficiency for a one-state MC. For a multistate MC, his algorithm produces arbitrarily long finite sequences. He does not, however, show how to paste these finite sequences together to produce *infinitely* long independent unbiased sequences." Blum [2] derived a beautiful algorithm to generate random bits from a degree-2 Markov chain in expected linear time by utilizing the von Neumann scheme for generating random bits from biased coin flips. While his approach can be extended to arbitrary out-degrees (the general Markov chain model used in this paper), the information-efficiency is still far from being optimal due to the low information-efficiency of the von Neumann scheme.

In this paper, we generalize Blum's algorithm to arbitrary degree finite Markov chains and combine it with existing methods for efficient generation of unbiased bits from biased coins, such as Elias's method. As a result, we provide the first known algorithm that generates unbiased random bits from arbitrary finite Markov chains, operates in expected linear time and achieves the information-theoretic upper bound on efficiency. Specifically, we propose an algorithm (that we call Algorithm A), that is a simple modification of Elias's suggestion to generate random bits, it operates on finite sequences and its efficiency

Manuscript received December 23, 2010; revised June 02, 2011; accepted October 04, 2011. Date of publication December 06, 2011; date of current version March 13, 2012. This work was supported in part by the National Science Foundation's Expeditions in Computing Program under Grant CCF-0832824. This paper was presented in part at the 2010 IEEE International Symposium on Information Theory.

The authors are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: hzhou@caltech.edu; bruck@caltech.edu).

Communicated by V. Guruswami, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2011.2175698

0018-9448/$26.00 © 2011 IEEE

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

can asymptotically reach the information-theoretic upper bound for long input sequences. In addition, we propose a second algorithm, called Algorithm B, that is a combination of Blum's and Elias's algorithms, it generates infinitely long sequences of random bits in expected linear time. One of our key ideas for generating random bits is that we explore equal-probability sequences of the same length. Hence, a natural question is: Can we improve the efficiency by utilizing as many as possible equal-probability sequences? We provide a positive answer to this question and describe Algorithm C, that is the first known polynomial-time and optimal algorithm (it is optimal in terms of information-efficiency for an arbitrary input length) for random bit generation from finite Markov chains.

In this paper, we use the following notations:

x_a	the ath element of X
$X[a]$	same as x_a, the ath element of X
$X[a:b]$	subsequence of X from the ath to bth element
X_a	$X[1:a]$
$X * Y$	the concatenation of X and Y e.g., $s_1s_2 * s_2s_1 = s_1s_2s_2s_1$
$Y \equiv X$	Y is a permutation of X e.g., $s_1s_2s_2s_3 \equiv s_3s_2s_2s_1$
$Y \doteq X$	Y is a permutation of X and $y_{\lvert Y\rvert} = x_{\lvert X\rvert}$ namely the last element is fixed e.g., $s_1s_2s_2s_3 \doteq s_2s_2s_1s_3$ where s_3 is fixed

The remainder of this paper is organized as follows. Section II reviews existing schemes for generating random bits from arbitrarily biased coins. Section III discusses the challenge in generating random bits from arbitrary finite Markov chains and presents our main lemma—this lemma characterizes the exit sequences of Markov chains. Algorithm A is presented and analyzed in Section IV, it is related to Elias's ideas for generating random bits from Markov chains. Algorithm B is presented in Section V, it is a generalization of Blum's algorithm. An optimal algorithm, called Algorithm C, is described in Section VI. Finally, Section VII provides numerical evaluations of our algorithms.

II. Generating Random Bits for Biased Coins

Consider a sequence of length N generated by a biased n-face coin

$$X = x_1x_2\dots x_N \in \{s_1, s_2, \dots, s_n\}^N$$

such that the probability to get s_i is p_i, and $\sum_{i=1}^{n} p_i = 1$. While we are given a sequence X the probabilities that $p_1, p_2, \dots, p_n$ are unknown, the question is: How can we efficiently generate an independent and unbiased sequence of 0's and 1's from X? The definition of efficiency for a generation algorithm is given as follows. This definition will be used throughout this paper.

Definition 1: Let X be a random sequence in $\{s_1, s_2, \dots, s_n\}^N$ and let $\Psi : \{s_1, s_2, \dots, s_n\}^N \to \{0,1\}^*$ be an algorithm generating random bits from X. Then given X, the efficiency (information-efficiency) of Ψ is defined as the ratio between the expected length of the output sequence and the length of the input sequence, i.e.

$$\eta = \frac{E[|\Psi(X)|]}{N}.$$

In this section, we describe three existing solutions for the problem of random bit generation from biased coins.

A. The von Neumann Scheme

In 1951, von Neumann [9] considered this question for biased coins and described a simple procedure for generating an independent unbiased binary sequence $z_1z_2\dots$ from the input sequence $X = x_1x_2\dots$. In his original procedure, the coin is binary, however, it can be simply generalized for the case of an n-face coin: For an input sequence, we can divide it into pairs $x_1x_2, x_3x_4, \dots$ and use the following mapping for each pair:

$$s_is_j(i<j) \to 0, \quad s_is_j(i>j) \to 1, \quad s_is_i \to \phi$$

where ϕ denotes the empty sequence. As a result, by concatenating the outputs of all the pairs, we can get a binary sequence which is independent and unbiased. The von Neumann scheme is computationally (very) fast, however, its information-efficiency is far from being optimal. For example, when the input sequence is binary, the probability for a pair of input bits to generate an output bit (not a ϕ) is $2p_1p_2$, hence the efficiency is p_1p_2, which is $\frac{1}{4}$ at $p_1 = p_2 = \frac{1}{2}$ and less elsewhere.

B. The Elias Scheme

In 1972, Elias [4] proposed an optimal (in terms of efficiency) algorithm as a generalization of the von Neumann scheme; for the sake of completeness we describe it here.

Elias's method is based on the following idea: The possible n^N input sequences of length N can be partitioned into classes such that all the sequences in the same class have the same number of s_k's for $1 \le k \le n$. Note that for every class, the members of the class have the same probability to be generated. For example, let $n = 2$ and $N = 4$, we can divide the possible $n^N = 16$ input sequences into 5 classes

$$\begin{aligned}
S_0 &= \{s_1s_1s_1s_1\}\\
S_1 &= \{s_1s_1s_1s_2, s_1s_1s_2s_1, s_1s_2s_1s_1, s_2s_1s_1s_1\}\\
S_2 &= \{s_1s_1s_2s_2, s_1s_2s_1s_2, s_1s_2s_2s_1,\\
&\qquad s_2s_1s_1s_2, s_2s_1s_2s_1, s_2s_2s_1s_1\}\\
S_3 &= \{s_1s_2s_2s_2, s_2s_1s_2s_2, s_2s_2s_1s_2, s_2s_2s_2s_1\}\\
S_4 &= \{s_2s_2s_2s_2\}.
\end{aligned}$$

Now, our goal is to assign a string of bits (the output) to each possible input sequence, such that any two output sequences Y and Y' with the same length (say k), have the same probability to be generated, namely $\frac{c_k}{2^n}$ for some $0 \le c_k \le 1$. The idea is that for any given class we partition the members of the class to sets of sizes that are a power of 2, for a set with 2^i members (for some i) we assign binary strings of length i. Note that when the class size is odd we have to exclude one member of this class. We now demonstrate the idea by continuing the example above.

Note that in the example above, we cannot assign any bits to the sequence in S_0, so if the input sequence is $s_1s_1s_1s_1$, the

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

output sequence should be ϕ (denoting the empty sequence). There are 4 sequences in S_1 and we assign the binary strings as follows:

$$s_1s_1s_1s_2 \to 00, \quad s_1s_1s_2s_1 \to 01$$
$$s_1s_2s_1s_1 \to 10, \quad s_2s_1s_1s_1 \to 11.$$

Similarly, for S_2, there are 6 sequences that can be divided into a set of 4 and a set of 2

$$s_1s_1s_2s_2 \to 00, \quad s_1s_2s_1s_2 \to 01$$
$$s_1s_2s_2s_1 \to 10, \quad s_2s_1s_1s_2 \to 11$$
$$s_2s_1s_2s_1 \to 0, \quad s_2s_2s_1s_1 \to 1.$$

In general, for a class with W members that were not assigned yet, assign 2^j possible output binary sequences of length j to 2^j distinct unassigned members, where $2^j \le W < 2^{j+1}$. Repeat the procedure above for the rest of the members that were not assigned. Note that when a class has an odd number of members, there will be one and only one member assigned to ϕ.

Given an input sequence X of length N, using the method above, the output sequence can be written as a function of X, denoted by $\Psi_E(X)$, called the Elias function. In [13], Ryabko and Matchikina showed that the Elias function of an input sequence of length N (that is generated by a biased coin with two faces) is computable in $O(N \log^3 N \log\log(N))$ time. We can prove that their conclusion is valid in the general case of a coin with n faces for any $n > 2$.

C. The Peres Scheme

In 1992, Peres [12] demonstrated that iterating the original von Neumann scheme on the discarded information can asymptotically achieve optimal efficiency. Let's define the function related to the von Neumann scheme as $\Psi_1 : \{0,1\}^* \to \{0,1\}^*$. Then the iterated procedures Ψ_v with $v \ge 2$ are defined inductively. Given input sequence $x_1x_2\ldots x_{2m}$, let $i_1 < i_2 < \ldots < i_k$ denote all the integers $i \le m$ for which $x_{2i} = x_{2i-1}$, then Ψ_v is defined as

$$\begin{aligned}&\Psi_v(x_1, x_2, \ldots, x_{2m})\\ &= \Psi_1(x_1, x_2, \ldots, x_{2m}) * \Psi_{v-1}(x_1 \oplus x_2, \ldots, x_{2m-1} \oplus x_{2m})\\ &\quad * \Psi_{v-1}(x_{2i_1}, \ldots, x_{2i_k}).\end{aligned}$$

Note that on the right-hand side (RHS) of the equation above, the first term corresponds to the random bits generated with the von Neumann scheme, the second and third terms relate to the symmetric information discarded by the von Neumann scheme. For example, when the input sequence is $X = 110100$, the output sequence based on the von Neumann scheme is

$$\Psi_1(110100) = 0$$

But based on the Peres scheme, we have the output sequence

$$\Psi_v(110100) = \Psi_1(110100) * \Psi_{v-1}(010) * \Psi_{v-1}(10)$$

which is 001, longer than that generated by the von Neumann Scheme.

Finally, we can define Ψ_v for sequences of odd length by

$$\Psi_v(x_1, x_2, \ldots, x_{2m+1}) = \Psi_v(x_1, x_2, \ldots, x_{2m}).$$

Surprisingly, this simple iterative procedure achieves the optimal efficiency asymptotically. The computational complexity and memory requirements of this scheme are substantially smaller than those of the Elias scheme. However, a drawback of this scheme is that its generalization to the case of an n-face coin with $n > 2$ is not obvious.

D. Properties of the Schemes

Let's denote $\Psi : \{s_1, s_2, \ldots, s_n\}^N \to \{0,1\}^*$ as a scheme that generates independent unbiased sequences from any biased coins (with unknown probabilities). Such Ψ can be the von Neumann scheme, the Elias scheme, the Peres scheme or any other scheme. Let X be a sequence generated from an arbitrary biased coin, with length N, then a property of Ψ is that for any $Y \in \{0,1\}^*$ and $Y' \in \{0,1\}^*$ with $|Y| = |Y'|$, we have

$$P[\Psi(X) = Y] = P[\Psi(X) = Y'].$$

Namely, two output sequences of equal length have equal probability.

This leads to the following property for Ψ. It says that given the number of s_i's for all i with $1 \le i \le n$, the number of such sequences yielding a binary sequence Y equals the number of such sequences yielding Y' when Y and Y' have the same length. It further implies that given the condition of knowing the number of s_i's for all i with $1 \le i \le n$, the output sequence of Ψ is still independent and unbiased. This property is due to the linear independence of probability functions of the sequences with different numbers of the s_i's.

Lemma 1: Let $S_{k_1,k_2,\ldots,k_n}$ be the subset of $\{s_1, s_2, \ldots, s_n\}^N$ consisting of all sequences with k_i appearances of s_i for all $1 \le i \le n$ such that $k_1 + k_2 + \cdots + k_n = N$. Let B_Y denote the set $\{X|\Psi(X) = Y\}$. Then for any $Y \in \{0,1\}^*$ and $Y' \in \{0,1\}^*$ with $|Y| = |Y'|$, we have

$$|S_{k_1,k_2,\ldots,k_n} \bigcap B_Y| = |S_{k_1,k_2,\ldots,k_n} \bigcap B_{Y'}|.$$

Proof: In $S_{k_1,k_2,\ldots,k_n}$, each sequence has k_i appearances of s_i for all $1 \le i \le n$. Given a biased coin with n faces and a sequence in $S_{k_1,k_2,\ldots,k_n}$, the probability of generating this sequence is

$$\beta_{k_1,k_2,\ldots,k_n}(p_1, p_2, \ldots, p_n) = \prod_{i=1}^{n} p_i^{k_i}$$

where p_i is the probability to get s_i with the biased coin.

Then the probability of generating Y is

$$\sum_{k_1+k_2+\cdots+k_n=N} |S_{k_1,\ldots,k_n} \bigcap B_Y| \beta_{k_1,\ldots,k_n}(p_1, \ldots, p_n).$$

And the probability of generating Y' is

$$\sum_{k_1+k_2+\cdots+k_n=N} |S_{k_1,\ldots,k_n} \bigcap B_{Y'}| \beta_{k_1,\ldots,k_n}(p_1, \ldots, p_n).$$

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

Since Ψ generates unbiased random sequences, we have $P[\Psi(X) = Y] = P[\Psi(X) = Y']$. As a result

$$\sum_{k_1+\cdots+k_n=N} (|S_{k_1,\ldots,k_n} \bigcap B_Y| - |S_{k_1,\ldots,k_n} \bigcap B'_Y|) \times \beta_{k_1,\ldots,k_n}(p_1,\ldots,p_n) = 0.$$

The set of polynomials

$$\bigcup_{k_1+\cdots+k_n=N} \{\beta_{k_1,\ldots,k_n}(p_1,\ldots,p_n)\}$$

is linearly independent in the vector space of functions on $\{(p_1,\ldots,p_n) \in [0,1]^n | p_1+p_2+\cdots+p_n = 1\}$, so we can conclude that $|S_{k_1,k_2,\ldots,k_n} \bigcap B_Y| = |S_{k_1,k_2,\ldots,k_n} \bigcap B_{Y'}|$. ■

III. Some Properties of Markov Chains

Our goal is to efficiently generate random bits from a Markov chain with unknown transition probabilities. The model we study is that a Markov chain generates the sequence of states that it is visiting and this sequence of states is the input sequence to our algorithm for generating random bits. Specifically, we express an input sequence as $X = x_1x_2\ldots x_N$ with $x_i \in \{s_1, s_2, \ldots, s_n\}$, where $\{s_1, s_2, \ldots, s_n\}$ indicate the states of a Markov chain.

One idea is that for a given Markov chain, we can treat each state, say s, as a coin and consider the 'next states' (the states the chain has transitioned to after being at state s) as the results of a coin toss. Namely, we can generate a collection of sequences $\pi(X) = [\pi_1(X), \pi_2(X), \ldots, \pi_n(X)]$, called exit sequences, where $\pi_i(X)$ is the sequence of states following s_i in X, namely

$$\pi_i(X) = \{x_{j+1} | x_j = s_i, 1 \le j < N\}.$$

For example, assume that the input sequence is

$$X = s_1s_4s_2s_1s_3s_2s_3s_1s_1s_2s_3s_4s_1$$

If we consider the states following s_1 we get $\pi_1(X)$ as the set of states in boldface:

$$X = s_1\mathbf{s_4}s_2s_1\mathbf{s_3}s_2s_3s_1\mathbf{s_1}\mathbf{s_2}s_3s_4s_1.$$

Hence, the exit sequences are

$$\pi_1(X) = s_4s_3s_1s_2$$
$$\pi_2(X) = s_1s_3s_3$$
$$\pi_3(X) = s_2s_1s_4$$
$$\pi_4(X) = s_2s_1.$$

Lemma 2 (Uniqueness): An input sequence X can be uniquely determined by x_1 and $\pi(X)$.

Proof: Given x_1 and $\pi(X)$, according to the work of Blum in [2], $x_1x_2\ldots x_N$ can uniquely be constructed in the following way: Initially, set the starting state as x_1. Inductively, if $x_i = s_k$, then set x_{i+1} as the first element in $\pi_k(X)$ and remove the first element of $\pi_k(X)$. Finally, we can uniquely generate the sequence $x_1x_2\ldots x_N$. ■

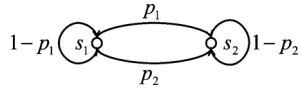

Fig. 1. An example of Markov chain with two states.

Lemma 3 (Equal-Probability): Two input sequences $X = x_1x_2\ldots x_N$ and $Y = y_1y_2\ldots y_N$ with $x_1 = y_1$ have the same probability to be generated if $\pi_i(X) \equiv \pi_i(Y)$ for all $1 \le i \le n$.

Proof: Note that the probability of generating X is

$$P[X] = P[x_1]P[x_2|x_1]\ldots P[x_N|x_{N-1}]$$

and the probability of generating Y is

$$P[Y] = P[y_1]P[y_2|y_1]\ldots P[y_N|y_{N-1}].$$

By permutating the terms in the expression above, it is not hard to get that $P[X] = P[Y]$ if $x_1 = y_1$ and $\pi_i(X) \equiv \pi_i(Y)$ for all $1 \le i \le n$. Basically, the exit sequences describe the edges that are used in the trajectory in the Markov chain. The edges in the trajectories that correspond to X and Y are identical, hence $P[X] = P[Y]$. ■

In [14], Samuelson considered a two-state Markov chain, and he pointed out that it may generate unbiased random bits by applying the von Neumann scheme to the exit sequence of state s_1. Later, in [4], in order to increase the efficiency, Elias has suggested a scheme that uses all the symbols produced by a Markov chain. His main idea was to create the final output sequence by concatenating the output sequences that correspond to $\pi_1(X), \pi_2(X), \ldots$. However, neither Samuelson nor Elias proved that their methods produce random output sequences that are independent and unbiased. In fact, their proposed methods are not correct for some cases. To demonstrate it we consider: 1) $\Psi(\pi_1(X))$ as the final output. 2) $\Psi(\pi_1(X)) * \Psi(\pi_2(X)) * \ldots$ as the final output. For example, consider the two-state Markov chain in which $P[s_2|s_1] = p_1$ and $P[s_1|s_2] = p_2$, as shown in Fig. 1.

Assume that an input sequence of length $N = 4$ is generated from this Markov chain and the starting state is s_1, then the probabilities of the possible input sequences and their corresponding output sequences are given in Table I. In the table we can see that the probabilities to produce 0 or 1 are different for some p_1 and p_2 in both methods, presented in columns 3 and 4, respectively.

The problem of generating random bits from an arbitrary Markov chain is challenging, as Blum said in [2]: "Elias's algorithm is excellent, but certain difficulties arise in trying to use it (or the original von Neumann scheme) to generate random bits in expected linear time from a Markov chain". It seems that the exit sequence of a state is independent since each exit of the state will not affect the other exits. However, this is not always true when the length of the input sequence is given, say N. Let's still consider the example of the two-state Markov chain in Fig. 1. Assume the starting state of this Markov chain is s_1, if $1 - p_1 > 0$, then with nonzero probability we have

$$\pi_1(X) = s_1s_1\ldots s_1$$

whose length is $N - 1$. But it is impossible to have

$$\pi_1(X) = s_2s_2\ldots s_2$$

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

TABLE I
PROBABILITIES OF EXIT SEQUENCES—AN EXAMPLE THAT SIMPLE CONCATENATION DOES NOT WORK

Input sequence	Probability	$\Psi(\pi_1(X))$	$\Psi(\pi_1(X)) * \Psi(\pi_2(X))$
$s_1s_1s_1s_1$	$(1-p_1)^3$	ϕ	ϕ
$s_1s_1s_1s_2$	$(1-p_1)^2p_1$	0	0
$s_1s_1s_2s_1$	$(1-p_1)p_1p_2$	0	0
$s_1s_1s_2s_2$	$(1-p_1)p_1(1-p_2)$	0	0
$s_1s_2s_1s_1$	$p_1p_2(1-p_1)$	1	1
$s_1s_2s_1s_2$	$p_1^2p_2$	ϕ	ϕ
$s_1s_2s_2s_1$	$p_1(1-p_2)p_2$	ϕ	1
$s_1s_2s_2s_2$	$p_1(1-p_2)^2$	ϕ	ϕ

of length $N-1$. That means $\pi_1(X)$ is not an independent sequence. The main reason is that although each exit of a state will not affect the other exits, it will affect the length of the exit sequence. In fact, $\pi_1(X)$ is an independent sequence if the length of $\pi_1(X)$ is given, instead of giving the length of X.

In this paper, we consider this problem from another perspective. According to Lemma 3, we know that permutating the exit sequences does not change the probability of a sequence, however, the permuted sequence has to correspond to a trajectory in the Markov chain. The reason for this contingency is that in some cases the permuted sequence does not correspond to a trajectory: Consider the following example:

$$X = s_1s_4s_2s_1s_3s_2s_3s_1s_1s_2s_3s_4s_1$$

and

$$\pi(X) = [s_4s_3s_1s_2, s_1s_3s_3, s_2s_1s_4, s_2s_1].$$

If we permute the last exit sequence s_2s_1 to s_1s_2, we cannot get a new sequence such that its starting state is s_1 and its exit sequences are

$$[s_4s_3s_1s_2, s_1s_3s_3, s_2s_1s_4, s_1s_2].$$

This can be verified by attempting to construct the sequence using Blum's method (which is given in the proof of Lemma 2). Notice that if we permute the first exit sequence $s_4s_3s_1s_2$ into $s_1s_2s_3s_4$, we *can* find such a new sequence, which is

$$Y = s_1s_1s_2s_1s_3s_2s_3s_1s_4s_2s_3s_4s_1.$$

This observation motivated us to study the characterization of exit sequences that are feasible in Markov chains (or finite state machines).

Definition 2 (Feasibility): Given a Markov chain, a starting state s_α and a collection of sequences $\Lambda = [\Lambda_1, \Lambda_2, \ldots, \Lambda_n]$, we say that (s_α, Λ) is feasible if and only if there exists a sequence X that corresponds to a trajectory in the Markov chain such that $x_1 = s_\alpha$ and $\pi(X) = \Lambda$.

Based on the definition of feasibility, we present the main technical lemma of the paper. Repeating the notation from the beginning of the paper, we say that a sequence Y is a tail-fixed permutation of X, denoted as $Y \doteq X$, if and only if: 1) Y is a permutation of X, and 2) X and Y have the same last element, namely, $y_{|Y|} = x_{|X|}$.

Lemma 4 (Main Lemma: Feasibility and Equivalence of Exit Sequences): Given a starting state s_α and two collections of sequences $\Lambda = [\Lambda_1, \Lambda_2, \ldots, \Lambda_n]$ and $\Gamma = [\Gamma_1, \Gamma_2, \ldots, \Gamma_n]$ such that $\Lambda_i \doteq \Gamma_i$ (tail-fixed permutation) for all $1 \le i \le n$. Then (s_α, Λ) is feasible if and only if (s_α, Γ) is feasible.

The proof of this main lemma will be given in Appendix. According to the main lemma, we have the following equivalent statement.

Lemma 5 (Feasible Permutations of Exit Sequences): Given an input sequence $X = x_1x_2\ldots x_N$ with $x_N = s_\chi$ that produced from a Markov chain. Assume that $[\Lambda_1, \Lambda_2, \ldots, \Lambda_n]$ is an aribitrary collection of exit sequences that corresponds to the exit sequences of X as follows:

1) Λ_i is a permutation ($\equiv$) of $\pi_i(X)$, for $i = \chi$.
2) Λ_i is a tail-fixed permutation ($\doteq$) of $\pi_i(X)$, for $i \neq \chi$.

Then there exists a feasible sequence $X' = x'_1x'_2\ldots x'_N$ such that $x'_1 = x_1$ and $\pi(X') = [\Lambda_1, \Lambda_2, \ldots, \Lambda_n]$. For this X', we have $x'_N = x_N$.

One might reason that Lemma 5 is stronger than the main lemma (Lemma 4). However, we will show that these two lemmas are equivalent. It is obvious that if the statement in Lemma 5 is true, then the main lemma is also true. Now we show that if the main lemma is true then the statement in Lemma 5 is also true.

Proof: Given $X = x_1x_2\ldots x_N$, let's add one more symbol s_{n+1} to the end of X (s_{n+1} is different from all the states in X), then we can get a new sequence $x_1x_2\ldots x_Ns_{n+1}$, whose exit sequences are

$$[\pi_1(X), \pi_2(X), \ldots, \pi_\chi(X)s_{n+1}, \ldots, \pi_n(X), \phi]$$

According to the main lemma, we know that there exists another sequence $x'_1x'_2\ldots x'_Nx'_{N+1}$ such that its exit sequences are

$$[\Lambda_1, \Lambda_2, \ldots, \Lambda_\chi s_{n+1}, \ldots, \Lambda_n, \phi]$$

and $x'_1 = x_1$. Definitely, the last symbol of this sequence is s_{n+1}, i.e., $x'_{N+1} = s_{n+1}$. As a result, we have $x'_N = s_\chi$.

Now, by removing the last element from $x'_1x'_2\ldots x'_Nx'_{N+1}$, we can get a new sequence $x = x'_1x'_2\ldots x'_N$ such that its exit sequences are

$$[\Lambda_1, \Lambda_2, \ldots, \Lambda_\chi, \ldots, \Lambda_n]$$

and $x'_1 = x_1$. We also have $x'_N = s_\chi$.

This completes the proof. ∎

We demonstrate the result above by considering the example at the beginning of this section. Let

$$X = s_1s_4s_2s_1s_3s_2s_3s_1s_1s_2s_3s_4s_1$$

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

with $\chi = 1$ and its exit sequences are given by

$$[s_4s_3s_1s_2, s_1s_3s_3, s_2s_1s_4, s_2s_1]$$

After permutating all the exit sequences (for $i \neq 1$, we keep the last element of the ith sequence fixed), we get a new group of exit sequences

$$[s_1s_2s_3s_4, s_3s_1s_3, s_1s_2s_4, s_2s_1].$$

Based on these new exit sequences, we can generate a new input sequence

$$X' = s_1s_1s_2s_3s_1s_3s_2s_1s_4s_2s_3s_4s_1$$

This accords with the statements above.

IV. Algorithm A: Modification of Elias's Suggestion

Elias suggested to generate random bits from an arbitrary Markov chain by concatenating the outputs of different exit sequences. In the above section, we showed that direct concatenation cannot always work. This motivates us to derive Algorithm A, which is a simple modification of Elias's suggestion and is able to generate random bits from any Markov chain efficiently.

Algorithm A

Input: A sequence $X = x_1x_2 \ldots x_N$ produced by a Markov chain, where $x_i \in S = \{s_1, s_2, \ldots, s_n\}$.

Output: A sequence Y of $0'$s and $1'$s.

Main Function:

Suppose $x_N = s_\chi$.

for $i := 1$ to n **do**

if $i = \chi$ **then**

Output $\Psi(\pi_i(X))$.

else

Output $\Psi(\pi_i(X)^{|\pi_i(X)|-1})$.

end if

end for

Comment: (1) $\Psi(X)$ can be any scheme that generates random bits from biased coins. For example, we can use the Elias function. (2) When $i = \chi$, we can also output $\Psi(\pi_i(X)^{|\pi_i(X)|-1})$ for simplicity, but the efficiency may be reduced a little.

The only difference between Algorithm A and direct concatenation is that: Algorithm A ignores the last symbols of some exit sequences. Let's go back to the example of a two-state Markov chain with $P[s_2|s_1] = p_1$ and $P[s_1|s_2] = p_2$ in Fig. 1, which demonstrates that direct concatenation does not always work well. Here, still assuming that an input sequence with length $N = 4$ is generated from this Markov chain with starting state s_1, then the probability of each possible input sequence and its corresponding output sequence (based on Algorithm A) are given by

Input sequence	Probability	Output sequence
$s_1s_1s_1s_1$	$(1-p_1)^3$	ϕ
$s_1s_1s_1s_2$	$(1-p_1)^2p_1$	ϕ
$s_1s_1s_2s_1$	$(1-p_1)p_1p_2$	0
$s_1s_1s_2s_2$	$(1-p_1)p_1(1-p_2)$	ϕ
$s_1s_2s_1s_1$	$p_1p_2(1-p_1)$	1
$s_1s_2s_1s_2$	$p_1^2p_2$	ϕ
$s_1s_2s_2s_1$	$p_1(1-p_2)p_2$	ϕ
$s_1s_2s_2s_2$	$p_1(1-p_2)^2$	ϕ

We can see that when the input sequence length $N = 4$, a bit 0 and a bit 1 have the same probability of being generated and no longer sequences are generated. In this case, the output sequence is independent and unbiased.

In order to prove that all the sequences generated by Algorithm A are independent and unbiased, we need to show that for any sequences Y and Y' of the same length, they have the same probability of being generated.

Theorem 6 (Algorithm A): Let the sequence generated by a Markov chain be used as input to Algorithm A, then the output of Algorithm A is an independent unbiased sequence.

Proof: Let's first divide all the possible sequences in $\{s_1, s_2, \ldots, s_n\}^N$ into classes, and use G to denote the set of the classes. Two sequences X and X' are in the same class if and only if

1) $x'_1 = x_1$ and $x'_N = x_N = s_\chi$ for some χ.
2) If $i = \chi$, $\pi_i(X') \equiv \pi_i(X)$.
3) If $i \neq \chi$, $\pi_i(X') \doteq \pi_i(X)$.

Let's use Ψ_A to denote Algorithm A. For $Y \in \{0,1\}^*$, let B_Y be the set of sequences X of length N such that $\Psi_A(X) = Y$. We show that for any $S \in G$, $|S \bigcap B_Y| = |S \bigcap B_{Y'}|$ whenever $|Y| = |Y'|$. If S is empty, this conclusion is trivial. In the following, we only consider the case that S is not empty.

Now, given a class S, if $i = \chi$ let's define S_i as the set consisting of all the permutations of $\pi_i(X)$ for $X \in S$, and if $i \neq \chi$ let's define S_i as the set consisting of all the permutations of $\pi_i(X)^{|\pi_i(X)|-1}$ for $X \in S$. For all $1 \leq i \leq n$ and $Y_i \in \{0,1\}^*$, we continue to define

$$S_i(Y_i) = \{\Lambda_i \in S_i | \Psi(\Lambda_i) = Y_i\}$$

which is the subset of S_i consisting of all sequences yielding Y_i. Based on Lemma 1, we know that $|S_i(Y_i)| = |S_i(Y'_i)|$ whenever $|Y_i| = |Y'_i|$. This implies that $|S_i(Y_i)|$ is a function of $|Y_i|$, which can be written as $M_i(|Y_i|)$.

For any partition of Y, namely $Y_1, Y_2, \ldots, Y_n$ such that $Y_1 * Y_2 * \ldots * Y_n = Y$, we have the following conclusion: $\forall \Lambda_1 \in S_1(Y_1), \Lambda_2 \in S_2(Y_2), \ldots, \Lambda_n \in S_n(Y_n)$, we can always find a sequence $X \in S \bigcap B_Y$ such that $\pi_i(X) = \Lambda_i$ for $i = \chi$ and $\pi_i(X)^{|\pi_i(X)|-1} = \Lambda_i$ for all $i \neq \chi$. This conclusion is immediate from Lemma 5. As a result, we have

$$|S \bigcap B_Y| = \sum_{Y_1 * Y_2 * \ldots * Y_n = Y} \prod_{i=1}^{n} |S_i(Y_i)|.$$

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

Let $l_1, l_2, \ldots, l_n$ be a group of nonnegative integers partitioning $|Y|$, then the formula above can be rewritten as

$$|S \bigcap B_Y| = \sum_{l_1+\cdots+l_n=|Y|} \prod_{i=1}^{n} M_i(l_i)$$

Similarly, we also have

$$|S \bigcap B_{Y'}| = \sum_{l_1+\cdots+l_n=|Y'|} \prod_{i=1}^{n} M_i(l_i)$$

which tells us that $|S \bigcap B_Y| = |S \bigcap B_{Y'}|$ if $|Y| = |Y'|$.

Note that all the sequences in the same class S have the same probability of being generated. So when $|Y| = |Y'|$, the probability of generating Y is

$$\begin{aligned} & P[X \in B_Y] \\ &= \sum_{S \in G} P[S] \sum_{X \in S} P[X \in B_Y | X \in S] \\ &= \sum_{S \in G} P[S] \sum_{X \in S} \frac{|S \bigcap B_Y|}{|S|} \\ &= \sum_{S \in G} P[S] \sum_{X \in S} \frac{|S \bigcap B_{Y'}|}{|S|} \\ &= P[X \in B_{Y'}] \end{aligned}$$

which implies that output sequence is independent and unbiased. ∎

Theorem 7 (Efficiency): Let X be a sequence of length N generated by a Markov chain, which is used as input to Algorithm A. Let Ψ in Algorithm A be Elias's function. Suppose the length of its output sequence is M, then the limiting efficiency $\eta_N = \frac{E[M]}{N}$ as $N \to \infty$ realizes the upper bound $\frac{H(X)}{N}$.

Proof: Here, the upper bound $\frac{H(X)}{N}$ is provided by Elias [4]. We can use the same argument in Elias's paper [4] to prove this theorem.

For all $1 \le i \le n$, let X_i denote the next state following s_i in the Markov chain. Then X_i is a random variable on $\{s_1, s_2, \ldots, s_n\}$ with distribution $\{p_{i1}, p_{i2}, \ldots, p_{\text{in}}\}$, where p_{ij} with $1 \le i, j \le n$ is the transition probability from state s_i to state s_j. The entropy of X_i is denoted as $H(X_i)$. Let $U = (u_1, u_2, \ldots, u_n)$ denote the stationary distribution of the Markov chain, then we have [3]

$$\lim_{N \to \infty} \frac{H(X)}{N} = \sum_{i=1}^{n} u_i H(X_i).$$

When $N \to \infty$, there exists an ϵ_N which $\to 0$, such that with probability $1 - \epsilon_N$, $|\pi_i(X)| > (u_i - \epsilon_N)N$ for all $1 \le i \le n$. Using Algorithm A, with probability $1 - \epsilon_N$, the length M of the output sequence is bounded below by

$$\sum_{i=1}^{n} (1 - \epsilon_N)(|\pi_i(X)| - 1)\eta_i$$

where η_i is the efficiency of the Ψ when the input is $\pi_i(X)$ or $\pi_i(X)^{|\pi_i(X)|-1}$. According to Theorem 2 in Elias's paper [4], we know that as $|\pi_i(X)| \to \infty$, $\eta_i \to H(X_i)$. So with probability $1 - \epsilon_N$, the length M of the output sequence is bounded from below by

$$\sum_{i=1}^{N} (1 - \epsilon_N)((u_i - \epsilon_N)N - 1)(1 - \epsilon_N)H(X_i).$$

Then we have

$$\begin{aligned} & \lim_{N \to \infty} \frac{E[M]}{N} \\ &\ge \lim_{N \to \infty} \frac{[\sum_{i=1}^{N} (1 - \epsilon_N)^3((u_i - \epsilon_N)N - 1)H(X_i)]}{N} \\ &= \lim_{N \to \infty} \frac{H(X)}{N}. \end{aligned}$$

At the same time, $\frac{E[M]}{N}$ is upper bounded by $\frac{H(X)}{N}$. So we can get

$$\lim_{N \to \infty} \frac{E[M]}{N} = \lim_{N \to \infty} \frac{H(X)}{N}$$

which completes the proof. ∎

Given an input sequence, it is efficient to generate independent unbiased sequences using Algorithm A. However, it has some limitations: 1) The complete input sequence has to be stored. 2) For a long input sequence it is computationally intensive as it depends on the input length. 3) The method works for finite-length sequences and does not lend itself to stream processing. In order to address these limitations we propose two variants of Algorithm A.

In the first variant of Algorithm A, instead of applying Ψ directly to $\Lambda_i = \pi_i(X)$ for $i = \chi$ (or $\Lambda_i = \pi_i(X)^{|\pi_i(X)|-1}$ for $i \ne \chi$), we first split Λ_i into several segments with lengths $k_{i1}, k_{i2}, \ldots$ then apply Ψ to all of the segments separately. It can be proved that this variant of Algorithm A can generate independent unbiased sequences from an arbitrary Markov chain, as long as $k_{i1}, k_{i2}, \ldots$ do not depend on the order of elements in each exit sequence. For example, we can split Λ_i into two segments of lengths $\lfloor \frac{|\Lambda_i|}{2} \rfloor$ and $\lceil \frac{|\Lambda_i|}{2} \rceil$, we can also split it into three segments of lengths $(a, a, |\Lambda_i| - 2a) \ldots$ Generally, the shorter each segment is, the faster we can obtain the final output. But at the same time, we may have to sacrifice a little information efficiency.

The second variant of Algorithm A is based on the following idea: for a given sequence from a Markov chain, we can split it into some shorter sequences such that they are independent of each other, therefore we can apply Algorithm A to all of the sequences and then concatenate their output sequences together as the final one. In order to do this, given a sequence $X = x_1 x_2 \ldots$, we can use $x_1 = s_\alpha$ as a special state to it. For example, in practice, we can set a constant k, if there exists a minimal integer i such that $x_i = s_\alpha$ and $i > k$, then we can split X into two sequences $x_1 x_2 \ldots x_i$ and $x_i x_{i+1} \ldots$ (note that both of the sequences have the element x_i). For the second sequence $x_i x_{i+1} \ldots$, we can repeat the same procedure ... Iteratively, we can split a sequence X into several sequences such that they are independent of each other. These sequences, with the exception

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

of the last one, start and end with s_α, and their lengths are usually slightly longer than k.

V. Algorithm B: Generalization of Blum's Algorithm

In [2], Blum proposed a beautiful algorithm to generate an independent unbiased sequence of 0's and 1's from any Markov chain by extending the von Neumann's scheme. His algorithm can deal with infinitely long sequences and uses only constant space and expected linear time. The only drawback of his algorithm is that its efficiency is still far from the information-theoretic upper bound, due to the limitation (compared to the Elias algorithm) of the von Neumann's scheme. In this section, we generalize Blum's algorithm by replacing von Neumann's scheme with Elias's. As a result, we get Algorithm B: It maintains some good properties of Blum's algorithm and its efficiency approaches the information-theoretic upper bound.

Algorithm B

Input: A sequence (or a stream) $x_1x_2\ldots$ produced by a Markov chain, where $x_i \in \{s_1, s_2, \ldots, s_n\}$.

Parameter: n positive integer functions (window size) $\varpi_i(k)$ with $k \geq 1$ for all $1 \leq i \leq n$.

Output: A sequence (or a stream) Y of 0's and 1's.

Main Function:

$E_i = \phi$ (empty) for all $1 \leq i \leq n$.

$k_i = 1$ for all $1 \leq i \leq n$.

c: the index of current state, namely, $s_c = x_1$.

while next input symbol is $s_j (\neq$ **null**$)$ **do**

$E_c = E_c s_j$ (Add s_j to E_c).

if $|E_j| \geq \varpi_j(k_j)$ **then**

Output $\Psi(E_j)$.

$E_j = \phi$.

$k_j = k_j + 1$.

end if

$c = j$.

end while

In the algorithm above, we apply function Ψ on E_j to generate random bits if and only if the window for E_j is completely filled and the Markov chain is currently at state s_j.

For example, we set $\varpi_i(k) = 4$ for all $1 \leq i \leq n$ and for all $k \geq 1$ and assume that the input sequence is

$$X = s_1s_1s_1s_2s_2s_2s_1s_2s_2.$$

After reading the last second (8th) symbol s_2, we have

$$E_1 = s_1s_1s_2s_2 \quad E_2 = s_2s_2s_1.$$

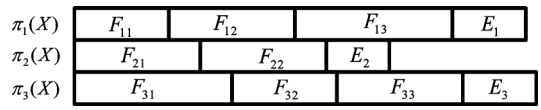

Fig. 2. The simplified expressions for the exit sequences of X.

In this case, $|E_1| \geq 4$ so the window for E_1 is full, but we don't apply Ψ to E_1 because the current state of the Markov chain is s_2, not s_1.

By reading the last (9th) symbol s_2, we get

$$E_1 = s_1s_1s_2s_2 \quad E_2 = s_2s_2s_1s_2$$

Since the current state of the Markov chain is s_2 and $|E_2| \geq 4$, we produce $\Psi(E_2 = s_2s_2s_1s_2)$ and reset E_2 as ϕ.

In the example above, treating X as input to Algorithm B, we get the output sequence $\Psi(s_2s_2s_1s_2)$. The algorithm does not output $\Psi(E_1 = s_1s_1s_2s_2)$ until the Markov chain reaches state s_1 again. Timing is crucial!

Note that Blum's algorithm is a special case of Algorithm B by setting the window size functions $\varpi_i(k) = 2$ for all $1 \leq i \leq n$ and $k \in \{1, 2, \ldots\}$. Namely, Algorithm B is a generalization of Blum's algorithm, the key is that when we increase the windows sizes, we can apply more efficient schemes (compared to the von Neumann's scheme) for Ψ. Assume a sequence of symbols $X = x_1x_2\ldots x_N$ with $x_N = s_\chi$ have been read by the algorithm above, we want to show that for any N, the output sequence is always independent and unbiased. Unfortunately, Blum's proof for the case of $\varpi_i(k) = 2$ cannot be applied to our proposed scheme.

For all i with $1 \leq i \leq n$, we can write

$$\pi_i(X) = F_{i1}F_{i2}\ldots F_{im_i}E_i$$

where F_{ij} with $1 \leq j \leq m_i$ are the segments used to generate outputs. For all i, j, we have

$$|F_{ij}| = \varpi_i(j)$$

and

$$\begin{cases} 0 \leq |E_i| < \varpi_i(m_i + 1) & \text{if } i = \chi \\ 0 < |E_i| \leq \varpi_i(m_i + 1) & \text{otherwise.} \end{cases}$$

See Fig. 2 for simple illustration.

Theorem 8 (Algorithm B): Let the sequence generated by a Markov chain be used as input to Algorithm B, then Algorithm B generates an independent unbiased sequence of bits in expected linear time.

Proof: In the following proof, we use the same idea as in the proof for Algorithm A.

Let's first divide all the possible input sequences in $\{s_1, s_2, \ldots, s_n\}^N$ into classes, and use G to denote the set consisting of all the classes. Two sequences X and X' are in the same class if and only if

1) $x_1 = x'_1$ and $x_N = x'_N$.
2) For all i with $1 \leq i \leq n$

$$\pi_i(X) = F_{i1}F_{i2}\ldots F_{im_i}E_i$$
$$\pi_i(X') = F'_{i1}F'_{i2}\ldots F'_{im_i}E'_i$$

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

where F_{ij} and F'_{ij} are the segments used to generate outputs.

3) For all i, j, $F_{ij} \equiv F'_{ij}$.
4) For all i, $E_i = E'_i$.

Let's use Ψ_B to denote Algorithm B. For $Y \in \{0,1\}^*$, let B_Y be the set of sequences X of length N such that $\Psi_B(X) = Y$. We show that for any $S \in G$, $|S \bigcap B_Y| = |S \bigcap B_{Y'}|$ whenever $|Y| = |Y'|$. If S is empty, this conclusion is trivial. In the following, we only consider the case that S is not empty.

Now, given a class S, let's define S_{ij} as the set consisting of all the permutations of F_{ij} for $X \in S$. Given $Y_{ij} \in \{0,1\}^*$, we continue to define

$$S_{ij}(Y_{ij}) = \{\Lambda_{ij} \in S_{ij} | \Psi(\Lambda_{ij}) = Y_{ij}\}$$

for all $1 \le i \le n$ and $1 \le j \le m_i$, which is the subset of S_{ij} consisting of all sequences yielding Y_{ij}. According to Lemma 1, we know that $|S_{ij}(Y_{ij})| = |S_{ij}(Y'_{ij})|$ whenever $|Y_{ij}| = |Y'_{ij}|$. This implies that $|S_{ij}(Y_{ij})|$ is a function of $|Y_{ij}|$, which can be written as $M_{ij}(|Y_{ij}|)$.

Let $l_{11}, l_{12}, \ldots, l_{1m_1}, l_{21} \ldots, l_{nm_n}$ be nonnegative integers such that their sum is $|Y|$, we want to prove that

$$|S \bigcap B_Y| = \sum_{l_{11}+\cdots+l_{nm_n}=|Y|} \prod_{i=1}^{n} \prod_{j=1}^{m_i} M_{ij}(l_{ij}).$$

The proof is by induction. Let $w = \sum_{i=1}^{n} m_i$. First, the conclusion holds for $w = 1$. Assume the conclusion holds for $w > 1$, we want to prove that the conclusion also holds for $w + 1$.

Note that for all $1 \le i \le n$, if $j_1 < j_2$, then F_{ij_1} generates an output before F_{ij_2} in Algorithm B. So given an input sequence $X \in S$, the last segment that generates an output (the output can be an empty string) is F_{im_i} for some i with $1 \le i \le n$. Now, we show that this i is fixed for all the sequences in S, i.e., the position of the last segment generating an output keeps unchanged. To prove this, given a sequence $X \in S$, let's see the first a symbols of X, i.e., X^a, such that the last segment F_{im_i} generates an output just after reading x_a when the input sequence is X. Based on our algorithm, X^a has the following properties.

1) The last symbol $x_a = s_i$.
2) $\pi_i(X^a) = F_{i1}F_{i2}\ldots F_{im_i}$.
3) $\pi_j(X^a) = F_{j1}F_{j2}\ldots F_{jm_j}\widetilde{E}_j$ for $j \ne i$, where $|\widetilde{E}_j| > 0$.

Now, let's permute each segment of $F_{11}, F_{12}, \ldots, F_{nm_n}$ to $F'_{11}, F'_{12}, \ldots, F'_{nm_n}$, then we get another sequence $X' \in S$. According to Lemma 5, if we consider the first a symbols of X', i.e., X'^a, it has the similar properties as X^a:

1) The last symbol $x'_a = s_i$.
2) $\pi_i(X'^a) = F'_{i1}F'_{i2}\ldots F'_{im_i}$.
3) $\pi_j(X'^a) = F'_{j1}F'_{j2}\ldots F'_{jm_j}\widetilde{E}_j$ for $j \ne i$, where $|\widetilde{E}_j| > 0$.

This implies that when the input sequence is X', F'_{im_i} generates an output just after reading x'_a and it is the last one. So we can conclude that for all the sequences in S, their last segments generating outputs are at the same position.

Let's fix the last segment F_{im_i} and assume F_{im_i} generates the last l_{im_i} bits of Y. We want to know how many sequences in $S \bigcap B_Y$ have F_{im_i} as their last segments that generate outputs? In order to get the answer, we concatenate F_{im_i} with E_i as the new E_i. As a result, we have $\sum_{i=1}^{n} m_i - 1 = w$ segments to generate the first $|Y| - l_{im_i}$ bits of Y. Based on our assumption, the number of such sequences will be

$$\sum_{l_{11}+\cdots+l_{i(m_i-1)}+\cdots=|Y|-l_{im_i}} \frac{1}{M_{im_i}(l_{im_i})} \prod_{k=1}^{n} \prod_{j=1}^{m_i} M_{kj}(l_{kj})$$

where $l_{11}, \ldots, l_{i(m_i-1)}, l_{(i+1)1} \ldots, l_{nm_n}$ are nonnegative integers. For each l_{im_i}, there are $M_{im_i}(l_{im_i})$ different choices for F_{im_i}. Therefore, $|S \bigcap B_Y|$ can be obtained by multiplying $M_{im_i}(l_{im_i})$ by the number above and summing them up over l_{im_i}. Namely, we can get the conclusion above.

According to this conclusion, we know that if $|Y| = |Y'|$, then $|S \bigcap B_Y| = |S \bigcap B_{Y'}|$. Using the same argument as in Theorem 6 we complete the proof of the theorem. ∎

Normally, the window size functions $\varpi_i(k)$ for $1 \le i \le n$ can be any positive integer functions. Here, we fix these window size functions as a constant, namely, ϖ. By increasing the value of ϖ, we can increase the efficiency of the scheme, but at the same time it may cost more storage space and need more waiting time. It is helpful to analyze the relationship between scheme efficiency and window size ϖ.

Theorem 9 (Efficiency): Let X be a sequence of length N generated by a Markov chain with transition matrix P, which is used as input to Algorithm B with constant window size ϖ. Then as the length of the sequence goes to infinity, the limiting efficiency of Algorithm B is

$$\eta(\varpi) = \sum_{i=1}^{n} u_i \eta_i(\varpi)$$

where $U = (u_1, u_2, \ldots, u_n)$ is the stationary distribution of this Markov chain, and $\eta_i(\varpi)$ is the efficiency of Ψ when the input sequence of length ϖ is generated by a n-face coin with distribution $(p_{i1}, p_{i2}, \ldots, p_{\mathrm{in}})$.

Proof: When $N \to \infty$, there exists an ϵ_N which $\to 0$, such that with probability $1 - \epsilon_N$, $(u_i - \epsilon_N)N < |\pi_i(X)| < (u_i + \epsilon_N)N$ for all $1 \le i \le n$.

The efficiency of Algorithm B can be written as $\eta(\varpi)$, which satisfies

$$\frac{\sum_{i=1}^{n} \lfloor \frac{|\pi_i(X)|-1}{\varpi} \rfloor \eta_i(\varpi)\varpi}{N} \le \eta(\varpi) \le \frac{\sum_{i=1}^{n} \lfloor \frac{|\pi_i(X)|}{\varpi} \rfloor \eta_i(\varpi)\varpi}{N}.$$

With probability $1 - \epsilon_N$, we have

$$\frac{\sum_{i=1}^{n} (\frac{(u_i-\epsilon_N)N}{\varpi} - 1)\eta_i(\varpi)\varpi}{N} \le \eta(\varpi) \le \frac{\sum_{i=1}^{n} \frac{(u_i-\epsilon_N)N}{\varpi} \eta_i(\varpi)\varpi}{N}.$$

So when $N \to \infty$, we have that

$$\eta(\varpi) = \sum_{i=1}^{n} u_i \eta_i(\varpi).$$

This completes the proof. ∎

Let's define $\alpha(N) = \sum n_k 2^{n_k}$, where $\sum 2^{n_k}$ is the standard binary expansion of N. Assume Ψ is the Elias function, then

$$\eta_i(\varpi) = \frac{1}{\varpi} \sum_{k_1+\cdots+k_n=\varpi} \alpha(\frac{\varpi!}{k_1!k_2!\ldots k_n!}) p_{i1}^{k_1} p_{i2}^{k_2} \ldots p_{\mathrm{in}}^{k_n}.$$

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

Based on this formula, we can numerically study the relationship between the limiting efficiency and the window size (see Section VII). In fact, when the window size becomes large, the limiting efficiency ($n \to \infty$) approaches the information-theoretic upper bound.

VI. Algorithm C: An Optimal Algorithm

Both Algorithm A and Algorithm B are asymptotically optimal, but when the length of the input sequence is finite they may not be optimal. In this section, we try to construct an optimal algorithm, called Algorithm C, such that its information-efficiency is maximized when the length of the input sequence is finite. Before presenting this algorithm, following the idea of Pae and Loui [11], we first discuss the equivalent condition for a function f to generate random bits from an arbitrary Markov chain, and then present the sufficient condition for f to be optimal.

Lemma 10 (Equivalent Condition): Let $K = \{k_{ij}\}$ be an $n \times n$ nonnegative integer matrix with $\sum_{i=1}^{n}\sum_{j=1}^{n} k_{ij} = N-1$. We define $S_{(\alpha,K)}$ as

$$S_{(\alpha,K)} = \{X \in \{s_1, s_2, \ldots, s_n\}^N | k_j(\pi_i(X)) = k_{ij}, x_1 = s_\alpha\}$$

where $k_j(X)$ is the number of s_j in X. A function $f : \{s_1, s_2, \ldots, s_n\}^N \to \{0,1\}^*$ can generate random bits from an arbitrary Markov chain, if and only if for any (α, K) and two binary sequences Y and Y' with $|Y| = |Y'|$

$$|S_{(\alpha,K)} \bigcap B_Y| = |S_{(\alpha,K)} \bigcap B_{Y'}|$$

where $B_Y = \{X | X \in \{s_1, s_2, \ldots, s_n\}^N, f(X) = Y\}$ is the set of sequences of length N that yield Y.

Proof: It is easy to see that if $|S_{(\alpha,K)} \bigcap B_Y| = |S_{(\alpha,K)} \bigcap B_{Y'}|$ for all (α, K) and $|Y| = |Y'|$, then Y and Y' have the same probability to be generated. In this case, f can generate random bits from an arbitrary Markov chain. In the rest, we only need to prove the inverse claim.

If f can generate random bits from an arbitrary Markov chain, then $P[f(X) = Y] = P[f(X) = Y']$ for any two binary sequences Y and Y' of the same length. Here, let p_{ij} be the transition probability from state s_i to state s_j for all $1 \le i, j \le n$, we can write

$$\begin{aligned} &P[f(X) = Y] \\ &= \sum_{\alpha,K \in G} |S_{(\alpha,K)} \bigcap B_Y| \phi_K(p_{11}, p_{12}, \ldots, p_{nn}) P(x_1 = s_\alpha) \end{aligned}$$

where

$$G = \{K | k_{ij} \in \{0\} \bigcup \mathbb{Z}^+, \sum_{i,j} k_{ij} = N-1\}$$

and

$$\phi_K(p_{11}, p_{12}, \ldots, p_{nn}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p_{ij}^{k_{ij}}.$$

Similarly

$$\begin{aligned} &P[f(X) = Y'] \\ &= \sum_{\alpha,K \in G} |S_{(\alpha,K)} \bigcap B_{Y'}| \phi_K(p_{11}, p_{12}, \ldots, p_{nn}) P(x_1 = s_\alpha). \end{aligned}$$

As a result

$$\begin{aligned} \sum_{\alpha,K \in G} (|S_{(\alpha,K)} \bigcap B_{Y'}| - |S_{(\alpha,K)} \bigcap B_Y|) \phi_K(p_{11}, \ldots, p_{nn}) \\ \times P(x_1 = s_\alpha) = 0. \end{aligned}$$

Since $P(x_1 = s_\alpha)$ can be any value in $[0,1]$, for all $1 \le \alpha \le n$ we have

$$\sum_{K \in G} (|S_{(\alpha,K)} \bigcap B_{Y'}| - |S_{(\alpha,K)} \bigcap B_Y|) \phi_K(p_{11}, \ldots, p_{nn}) = 0.$$

It can be proved that $\bigcup_{K \in G} \{\phi_K(p_{11}, p_{12}, \ldots, p_{nn})\}$ are linear independent in the vector space of functions on the transition probabilities, namely

$$\{(p_{11}, p_{12}, \ldots, p_{nn}) | p_{ij} \in [0,1], \sum_{j=1}^{n} p_{ij} = 1\}.$$

Based on this fact, we can conclude that $|S_{(\alpha,K)} \bigcap B_Y| = |S_{(\alpha,K)} \bigcap B_{Y'}|$ for all (α, K) if $|Y| = |Y'|$. ∎

Let's define $\alpha(N) = \sum n_k 2^{n_k}$, where $\sum 2^{n_k}$ is the standard binary expansion of N, then we have the sufficient condition for an optimal function.

Lemma 11 (Sufficient Condition for an Optimal Function): Let f^* be a function that generates random bits from an arbitrary Markov chain with unknown transition probabilities. If for any α and any $n \times n$ non-negative integer matrix K with $\sum_{i=1}^{n}\sum_{j=1}^{n} k_{ij} = N-1$, the following equation is satisfied,

$$\sum_{X \in S_{(\alpha,K)}} |f^*(X)| = \alpha(|S_{(\alpha,K)}|)$$

then f^* generates independent unbiased random bits with optimal information efficiency. Note that $|f^*(X)|$ is the length of $f^*(x)$ and $|S_{(\alpha,K)}|$ is the size of $S_{(\alpha,K)}$.

Proof: Let h denote an arbitrary function that is able to generate random bits from any Markov chain. According to [11, Lemma 2.9], we know that

$$\sum_{X \in S_{(\alpha,K)}} |h(X)| \le \alpha(|S_{(\alpha,K)}|).$$

Then the average output length of h is

$$\begin{aligned} E(|h(X)|) &= \frac{1}{N} \sum_{(\alpha,K)} \sum_{X \in S_{(\alpha,K)}} |h(X)| \phi(K) P[x_1 = s_\alpha] \\ &\le \frac{1}{N} \sum_{(\alpha,K)} \alpha(|S_{(\alpha,K)}|) \phi(K) P[x_1 = s_\alpha] \\ &= \frac{1}{N} \sum_{(\alpha,K)} \sum_{X \in S_{(\alpha,K)}} |f^*(X)| \phi(K) P[x_1 = s_\alpha] \\ &= E(|f^*(X)|). \end{aligned}$$

So f^* is the optimal one. This completes the proof. ∎

Here, we construct the following algorithm (Algorithm C) which satisfies all the conditions in Lemma 10 and Lemma 11.

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

As a result, it can generate unbiased random bits from an arbitrary Markov chain with optimal information efficiency.

Algorithm C

Input: A sequence $X = x_1x_2\ldots,x_N$ produced by a Markov chain, where $x_i \in S = \{s_1, s_2, \ldots, s_n\}$.
Output: A sequence Y of $0'$s and $1'$s.
Main Function:
1) Get the matrix $K = \{k_{ij}\}$ with

$$k_{ij} = k_j(\pi_i(X)).$$

2) Define $S(X)$ as

$$S(X) = \{X' | k_j(\pi_i(X')) = k_{ij} \forall i, j; x'_1 = x_1\}$$

then compute $|S(X)|$.
3) Compute the rank $r(X)$ of X in $S(X)$ with respect to a given order. The rank with respect to a lexicographic order will be given later.
4) According to $|S(X)|$ and $r(X)$, determine the output sequence. Let $\sum_k 2^{n_k}$ be the standard binary expansion of $|S(X)|$ with $n_1 > n_2 > \ldots$ and assume the starting value of $r(X)$ is 0. If $r(X) < 2^{n_1}$, the output is the n_1 digit binary representation of $r(x)$. If $\sum_{k=1}^{i} 2^{n_k} \le r(x) < \sum_{k=1}^{i+1} 2^{n_k}$, the output is the n_{i+1} digit binary representation of $r(x)$.
Comment: The fast calculations of $|S(X)|$ and $r(x)$ will be given in the rest of this section.

In Algorithm A, when we use Elias's function as Ψ, the limiting efficiency $\eta_N = \frac{E[M]}{N}$ (as $N \to \infty$) realizes the bound $\frac{H(X)}{N}$. Algorithm C is optimal, so it has the same or higher efficiency. Therefore, the limiting efficiency of Algorithm C as $N \to \infty$ also realizes the bound $\frac{H(X)}{N}$.

In Algorithm C, for an input sequence X with $x_N = s_\chi$, we can rank it with respect to the lexicographic order of $\theta(X)$ and $\sigma(X)$. Here, we define

$$\theta(X) = (\pi_1(X)_{|\pi_1(X)|}, \ldots, \pi_n(X)_{|\pi_n(X)|})$$

which is the vector of the last symbols of $\pi_i(X)$ for $1 \le i \le n$. And $\sigma(X)$ is the complement of $\theta(X)$ in $\pi(X)$, namely

$$\sigma(X) = (\pi_1(X)^{|\pi_1(X)|-1}, \ldots, \pi_n(X)^{|\pi_n(X)|-1}).$$

For example, when the input sequence is

$$X = s_1s_4s_2s_1s_3s_2s_3s_1s_1s_2s_3s_4s_1$$

Its exit sequences are

$$\pi(X) = [s_4s_3s_1s_2, s_1s_3s_3, s_2s_1s_4, s_2s_1]$$

Then for this input sequence X, we have that

$$\theta(X) = [s_2, s_3, s_4, s_1]$$
$$\sigma(X) = [s_4s_2s_1, s_1s_3, s_2s_1, s_2]$$

Based on the lexicographic order defined above, both $|S(X)|$ and $r(X)$ can be obtained using a brute-force search. However, this approach in not computationally efficient. Here, we describe an efficient algorithm for computing $|S(X)|$ and $r(X)$ when n is a small constant, such that Algorithm C is computable in $O(N \log^3 N \log\log N)$ time. This method is inspired by the algorithm for computing the Elias function that is described in [13]. However, when n is not small, the complexity of computing $|S(X)|$ (or $r(x)$) has an exponential dependence on n, which will make this algorithm much slower in computation than the previous algorithms.

Lemma 12: Let

$$Z = \left(\prod_{i=1}^{n} \frac{(k_{i1} + k_{i2} + \cdots + k_{\mathrm{in}})!}{k_{i1}!k_{i2}!\ldots k_{\mathrm{in}}!}\right)$$

and let $N = \sum_{i=1}^{n}\sum_{j=1}^{n} k_{ij}$, then Z is computable in $O(N \log^3 N \log\log N)$ time (not related with n).

Proof: It is known that given two numbers of length n bits, their multiplication or division is computable in $O(n \log n \log\log n)$ time based on Schönhage-Strassen algorithm [1]. We can calculate Z based on this fast multiplication.

For simplification, we denote $k_i = \sum_{j=1}^{n} k_{ij}$. Note that we can write Z as a multiplication of N terms, namely

$$\frac{k_1}{1}, \frac{k_1}{2}, \ldots, \frac{k_1}{k_{11}}, \frac{k_1}{1}, \frac{k_1}{2} \ldots, \frac{k_n}{k_{nn}}$$

which are denoted as

$$\rho_1^0, \rho_2^0, \ldots, \rho_{N-1}^0, \rho_N^0.$$

It is easy to see that the notation of every ρ_i^0 used $2\log_2 N$ bits ($\log_2 N$ for the numerator and $\log N$ for the denominator). The total time to compute all of them is much less than $O(N \log^3 N \log\log N)$.

Based on these notations, we write Z as

$$Z = \rho_1^0\rho_2^0\ldots\rho_{N-1}^0\rho_N^0$$

Suppose that $\log_2 N$ is an integer. Otherwise, we can add trivial terms to the formula above to make $\log_2 N$ be an integer. In order to calculate Z quickly, the following calculations are performed:

$$\rho_i^s = \rho_{2i-1}^{s-1}\rho_{2i}^{s-1}$$
$$s = 1, 2, \ldots, \log_2 N; \quad i = 1, 2, \ldots, 2^{-s}N$$

Then we are able to compute Z iteratively and finally get

$$Z = \rho_1^{\log_2 N}$$

To calculate ρ_i^1 for $i = 1, 2, \ldots, N/2$, it takes $2(N/2)$ multiplications of numbers with length $\log_2 N$ bits. Similarly, to calculate ρ_i^s for $i = 1, 2, \ldots, N/2$, it takes $2(N/2^s)$ multiplications of numbers with length $2^s \log_2 N$ bits. So the time complexity of computing Z is

$$\sum_{s=1}^{\log_2 N} 2(N/2^s)O(2^s \log_2 N \log(2^s \log_2 N) \log\log(2^s \log_2 N)).$$

This value is not greater than

$$O(N \log^2 N \log(N \log N) \log\log(N \log N))$$

which yields the result in the lemma. ∎

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

Lemma 13: Let n be a small constant, then $|S(X)|$ in Algorithm C is computable in $O(N\log^3 N\log\log N)$ time.

Proof: The idea to compute $|S(X)|$ in Algorithm C is that we can divide $S(X)$ into different classes, denoted by $S(X,\theta)$ for different θ such that

$$S(X,\theta)=\{X'|\forall i,j,k_j(\pi_i(X'))=k_{ij},\theta(X')=\theta\}$$

where $k_{ij}=k_j(\pi_i(X))$ is the number of s_j's in $\pi_i(X)$ for all $1\le i,j\le n$. $\theta(X)$ is the vector of the last symbols of $\pi(X)$ defined above. As a result, we have $|S(X)|=\sum_\theta|S(X,\theta)|$. Although it is not easy to calculate $|S(X)|$ directly, but it is much easier to compute $|S(X,\theta)|$ for a given θ.

For a given $\theta=(\theta_1,\theta_2,\ldots,\theta_n)$, we need first determine whether $S(X,\theta)$ is empty or not. In order to do this, we quickly construct a collection of exit sequences $\Lambda=[\Lambda_1,\Lambda_2,\ldots,\Lambda_n]$ by moving the first θ_i in $\pi_i(X)$ to the end for all $1\le i\le n$. According to the main lemma, we know that $S(X,\theta)$ is empty if and only if $\pi_i(X)$ does not include θ_i for some i or (x_1,Λ) is not feasible.

If $S(X,\theta)$ is not empty, then (x_1,Λ) is feasible. In this case, based on the main lemma, we have

$$|S(X,\theta)|=\prod_{i=1}^{n}\frac{(k_{i1}+k_{i2}+\cdots+k_{\rm in}-1)!}{k_{i1}!\ldots(k_{i\theta_i}-1)!\ldots k_{\rm in}!}$$
$$=\Big(\prod_{i=1}^{n}\frac{(k_{i1}+k_{i2}+\cdots+k_{\rm in})!}{k_{i1}!k_{i2}!\ldots k_{\rm in}!}\Big)\Big(\prod_{i=1}^{n}\frac{k_{i\theta_i}}{(k_{i1}+k_{i2}+\cdots+k_{\rm in})}\Big)$$

Here, we let

$$Z=\Big(\prod_{i=1}^{n}\frac{(k_{i1}+k_{i2}+\cdots+k_{\rm in})!}{k_{i1}!k_{i2}!\ldots k_{\rm in}!}\Big).$$

Then we can get

$$|S(X)|=\sum_\theta|S(X,\theta)|=Z\Big(\sum_\theta\prod_{i=1}^{n}\frac{k_{i\theta_i}}{(k_{i1}+k_{i2}+\cdots+k_{\rm in})}\Big).$$

According to Lemma 12, Z is computable in $O(N\log^3 N\log\log N)$ time. So if n is a small constant, then $|S(X)|$ is also computable in $O(N\log^3 N\log\log N)$ time. However, when n is not small, we have to enumerate all the possible combinations for θ with $O(n^n)$ time, which is not computationally efficient. ■

Lemma 14: Let n be a small constant, then $r(X)$ in Algorithm C is computable in $O(N\log^3 N\log\log N)$ time.

Proof: Based on some calculations in the lemma above, we can try to obtain $r(X)$ when X is ranked with respect to the lexicographic order of $\theta(X)$ and $\sigma(X)$. Let $r(X,\theta(X))$ denote the rank of X in $S(X,\theta(X))$, then we have that

$$r(X)=\sum_{\theta<\theta(X)}|S(X,\theta)|+r(X,\theta(X))$$

where $<$ is based on the lexicographic order. In the formula, when n is a small constant, $\sum_{\theta<\theta(X)}|S(X,\theta)|$ can be obtained in $O(N\log^3 N\log\log N)$ time by computing

$$Z\frac{\sum_{\theta<\theta(X):|S(X,\theta)|>0}\prod_{i=1}^{n}k_{i\theta_i}}{\prod_{i=1}^{n}(k_{i1}+k_{i2}+\cdots+k_{\rm in})}$$

where Z is defined in the last lemma and the second term can be calculated fast when n is a small constant. (However, n cannot be big, since the complexity of computing the second term has an exponential dependence on n.)

So far, we only need to compute $r(X,\theta(X))$, with respect to the lexicography order of $\sigma(X)$. Here, we write $\sigma(X)$ as the concatenation of a group of sequences, namely

$$\sigma(X)=\sigma_1(X)*\sigma_2(X)*\ldots*\sigma_n(X)$$

such that for all $1\le i\le n$ $\sigma_i(X)=\pi_i(X)^{|\pi_i(X)|-1}$.

There are $M=(N-1)-n$ symbols in $\sigma(X)$. Let $r_i(X)$ be the number of sequences in $S(X,\theta(X))$ such that their first $M-i$ symbols are $\sigma(X)[1,M-i]$ and their $M-i+1^{\rm th}$ symbols are smaller than $\sigma(X)[M-i+1]$. Then we can get that

$$r(X,\theta(X))=\sum_{i=1}^{M}r_i(X)$$

Let's assume that $\sigma(X)[M-i+1]=s_{w_i}$ for some w_i, and it is the $u_i^{\rm th}$ symbol in $\sigma_{v_i}(X)$. For simplicity, we denote $\sigma_{v_i}(X)[u_i,|\sigma_{v_i}(X)|]$ as ζ_i. For example, when $n=3$ and $[\sigma_1(X),\sigma_2(X),\sigma_3(X)]=[s_1s_2,s_2s_3,s_1s_1s_1]$, we have

$$\zeta_1=s_1,\zeta_2=s_1s_1,\zeta_3=s_1s_1s_1,\zeta_4=s_3,\zeta_5=s_2s_3,\ldots.$$

To calculate $r_i(X)$, we can count all the sequences generated by permuting the symbols of $\zeta_i,\sigma_{v_i+1}(X),\ldots,\sigma_n(X)$ such that the $M-i+1^{\rm th}$ symbol of the new sequence is smaller than s_{w_i}. Then we can get

$$r_i(X)=\sum_{j<w_i}\frac{(|\zeta_i|-1)!}{k_1(\zeta_i)!\ldots(k_j(\zeta_i)-1)!\ldots k_n(\zeta_i)!}$$
$$\times\prod_{i=v_i+1}^{n}\frac{|\sigma_i(X)|!}{k_1(\sigma_i(X))!k_2(\sigma_i(X))!\ldots k_n(\sigma_i(X))!}$$

where $k_j(X)$ counts the number of s_i's in X.

Let's define the values

$$\rho_{i-1}^0=\frac{|\zeta_i|}{k_{w_i}(\zeta_i)}$$

for all $1\le i\le M$. In this expression, $k_{w_i}(\zeta_i)$ is the number of s_{w_i}'s in ζ_i, and s_{w_i} is the first symbol of ζ_i.

It is easy to show that for $1\le i\le M$

$$\rho_{i-1}^0\rho_{i-2}^0\cdots\rho_2^0\rho_1^0=\frac{|\zeta_i|!}{k_1(\zeta_i)!\ldots k_j(\zeta_i)!\ldots k_n(\zeta_i)!}$$
$$\times\prod_{i=v+1}^{n}\frac{|\sigma_i(X)|!}{k_1(\sigma_i(X))!k_2(\sigma_i(X))!\ldots k_n(\sigma_i(X))!}$$

If we also define the values

$$\lambda_i^0=\frac{\sum_{j<w_i}k_j(\zeta_i)}{|\zeta_i|}$$

for all $1\le i\le M$, then we have

$$r_i(X)=\lambda_i^0\rho_{i-1}^0\rho_{i-2}^0\ldots\rho_1^0$$

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

and

$$r(X,\theta(X)) = \sum_{i=1}^{M} \lambda_i^0 \rho_{i-1}^0 \rho_{i-2}^0 \cdots \rho_2^0 \rho_1^0.$$

Suppose that $\log_2 M$ is an integer. Otherwise, we can add trivial terms to the formula above to make $\log_2 M$ an integer. In order to quickly calculate $r(X,\theta(X))$, the following calculations are performed for s from 1 to $\log_2 M$:

$$\rho_i^s = \rho_{2i}^{s-1}\rho_{2i-1}^{s-1}, \quad i = 1,2,\ldots,2^{-s}M-1$$
$$\lambda_i^s = \lambda_{2i-1}^{s-1} + \lambda_{2i}^{s-1}\rho_{2i-1}^{s-1} \quad i = 1,2,\ldots,2^{-s}M.$$

By computing all ρ_i^s and λ_i^s for s from 1 to $\log_2 M$ iteratively, we can get that

$$r(X,\theta(X)) = \lambda_1^{\log_2 M}.$$

Now, we use the same idea in [13] to analyze the computational complexity. Note that every ρ_i^0 and λ_i^0 can be represented using $2\log_2 M$ bits ($\log_2 M$ for the numerator and $\log M$ for the denominator). And we can calculate all of them quickly. To calculate ρ_i^1 for $i = 1,2,\ldots,M/2-1$, it takes at most $2(M/2)$ multiplications of numbers with length $\log_2 M$ bits. To calculate λ_i^1 for $i = 1,2,\ldots,M/2$, it takes $3(M/2)$ multiplications of numbers with length $\log_2 M$ bits. That is because we can write λ_i^1 as $\frac{a}{b}+\frac{c}{d} = \frac{ad+bc}{bd}$ for some integers a, b, c, d with length $\log_2 M$ bits. Similarly, to calculate all ρ_i^s and λ_i^s for some s, it takes at most $5(M/2^s)$ multiplications of numbers with length $2^s \log_2 M$ bits. As a result, the time complexity of computing Z is

$$\sum_{s=1}^{\log_2 M} 5(M/2^s) O(2^s \log_2 M \log(2^s \log_2 M) \log\log(2^s \log_2 M))$$

which is computable in $O(M\log^3 M \log\log M)$ time. As a result, for a small constant n, $r(X)$ is computable in $O(N\log^3 N\log\log N)$ time. ∎

Based on the discussion above, we know that Algorithm C is computable in $O(N\log^3 N\log\log N)$ time when n is a small constant. However, when n is not a constant, this algorithm is not computationally efficient since its time complexity depends exponentially on n.

VII. Numerical Results

In this section, we describe numerical results related to the implementations of Algorithm A, Algorithm B, and Algorithm C. We use the Elias function for Ψ.

In the first experiment, we use the following randomly generated transition matrix for a Markov chain with three states

$$P = \begin{pmatrix} 0.300987 & 0.468876 & 0.230135 \\ 0.462996 & 0.480767 & 0.056236 \\ 0.42424 & 0.032404 & 0.543355 \end{pmatrix}.$$

Consider a sequence of length 12 that is generated by the Markov chain defined above and assume that s_1 is the first state of this sequence. Namely, there are $3^{11} = 177147$ possible input sequences. For each possible input sequence, we can compute its generating probability and the corresponding output sequences using our three algorithms. Table II presents the results of calculating the probabilities of all possible output sequences for the three algorithms. Note that the results show that indeed the outputs of the algorithms are independent unbiased sequences. Also, Algorithm C has the highest information efficiency (it is optimal), and Algorithm A has a higher information efficiency than Algorithm B (with window size 4).

TABLE II
The Probability of Each Possible Output Sequence and the Expected Output Length

Output	Probability Algorithm A	Probability Algorithm B with $\varpi = 4$	Probability Algorithm C
Λ	0.0224191	0.1094849	0.0208336
0	0.0260692	0.0215901	0.0200917
1	0.0260692	0.0215901	0.0200917
00	0.0298179	0.1011625	0.0206147
10	0.0298179	0.1011625	0.0206147
01	0.0298179	0.1011625	0.0206147
11	0.0298179	0.1011625	0.0206147
000	0.0244406	0.0242258	0.0171941
100	0.0244406	0.0242258	0.0171941
...	...	...	...
011111	0.0018831	1.39E-5	0.0029596
111111	0.0018831	1.39E-5	0.0029596
0000000	1.305E-4		6.056E-4
1000000	1.305E-4		6.056E-4
...			...
0111111	1.305E-4		6.056E-4
1111111	1.305E-4		6.056E-4
00000000			1.44E-5
10000000			1.44E-5
...			...
01111111			1.44E-5
11111111			1.44E-5
Expected Length	3.829	2.494	4.355

In the second calculation, we want to test the influence of window size ϖ (assume $\varpi_i(k) = \varpi$ for $1 \le i \le n$) on the efficiency of Algorithm B. Since the efficiency depends on the transition matrix of the Markov chain we decided to evaluate of the efficiency related to the uniform transition matrix, namely all the entries are $\frac{1}{n}$, where n is the number of states. We assume that n is infinitely large. In this case, the stationary distribution of the Markov chain is $\{\frac{1}{n},\frac{1}{n},\ldots,\frac{1}{n}\}$. Fig. 3 shows that when $\varpi = 2$ (Blum's Algorithm), the limiting efficiencies for $n = (2,3,5)$ are $(\frac{1}{4},\frac{1}{3},\frac{2}{5})$, respectively. When $\varpi = 15$, their corresponding efficiencies are $(0.7228, 1.1342, 1.5827)$. So if the input sequence is long enough, by changing ϖ from 2 to 15, the efficiency can increase 189% for $n = 2$, 240% for $n = 3$ and 296% for $n = 4$. When ϖ is small, we can increase the efficiency of Algorithm B significantly by increasing the window size ϖ. When ϖ becomes larger, the efficiency of Algorithm B will converge to the information-theoretical upper bound, namely, $\log_2 n$. Note that 3 is not a good value for the window size in the algorithm. That is because the Elias function is not very efficient when the length of the input sequence is 3. Let's consider a biased coin with two states s_1, s_2. If the input sequence is $s_1s_1s_1$ or $s_2s_2s_2$, the Elias function will generate

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

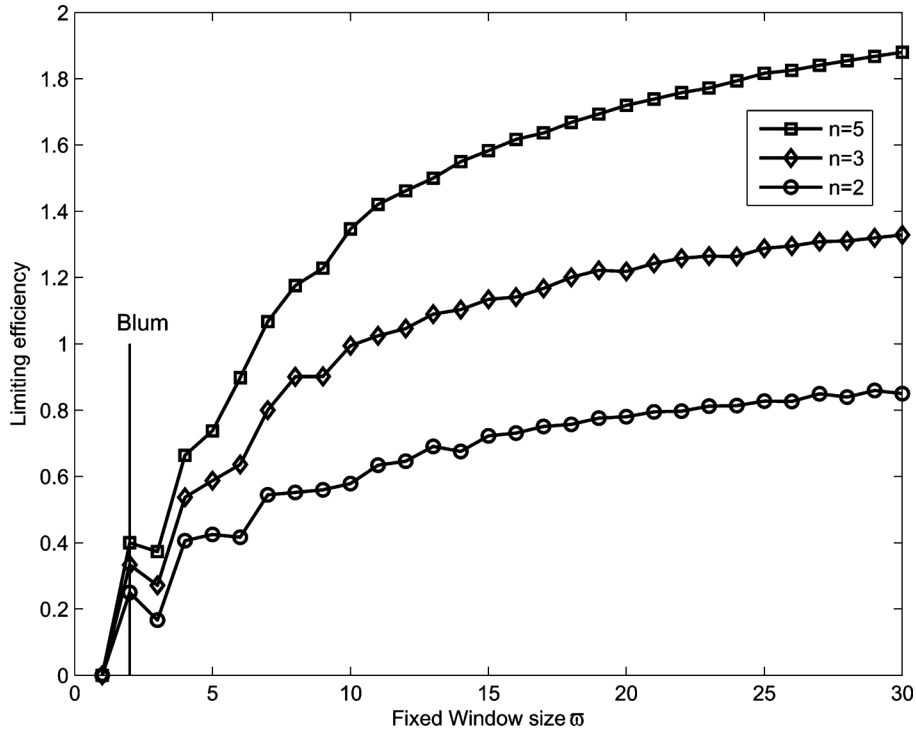

Fig. 3. The limiting efficiency of Algorithm B varies with the value of window size ϖ for different state number n, where we assume that the transition probability $p_{ij} = \frac{1}{n}$ for all $1 \leq i, j \leq n$.

nothing. For all other cases, it has only $2/3$ chance to generate one bit and $1/3$ chance to generate nothing. As a result, the efficiency is even worse than the efficiency when the length of the input sequence equals 2.

VIII. Concluding Remarks

We considered the classical problem of generating independent unbiased bits from an arbitrary Markov chain with unknown transition probabilities. Our main contribution is the first known algorithm that has expected linear time complexity and achieves the information-theoretic upper bound on efficiency.

Our work is related to a number of interesting results in both computer science and information theory. In computer science, the attention has focused on extracting randomness from a general weak random source (introduced by Zuckerman [18]). Hence, the concept of an extractor was introduced—it uses a small number of truly random bits as the seed (catalyst) for randomness extraction. During the past two decades, extractors and their applications have been studied extensively, see [10], [15] for surveys on the topic. While our algorithms generate truly random bits (given a prefect Markov chain as a source) the goal of extractors is to generate random sequences which are asymptotically close to random bits in the sense of statistical distance.

In information theory, it was discovered that optimal source codes can be used as universal random bit generators from arbitrary stationary ergodic random sources [16], [6] (Markov chains studied in our paper are special cases of stationary ergodic sources). When the input sequence is generated from a stationary ergodic process and it is long enough one can obtain an output sequence that behaves like truly random bits in the sense of normalized divergence. However, in some cases, the definition of normalized divergence is not strong enough. For example, suppose Y is a sequence of unbiased random bits in the sense of normalized divergence, and $1 * Y$ is Y with a 1 concatenated at the beginning. If the sequence Y is long enough the sequence $1 * Y$ is a sequence of unbiased random bits in the sense of normalized divergence. However the sequence $1 * Y$ might not be useful in applications that are sensitive to the randomness of the first bit.

Appendix

In this appendix, we prove the main lemma.

Lemma 4 (Main Lemma: Feasibility and Equivalence of Exit Sequences): Given a starting state s_α and two collections of sequences $\Lambda = [\Lambda_1, \Lambda_2, \ldots, \Lambda_n]$ and $\Gamma = [\Gamma_1, \Gamma_2, \ldots, \Gamma_n]$ such that $\Lambda_i \doteq \Gamma_i$ (tail-fixed permutation) for all $1 \leq i \leq n$. Then (s_α, Λ) is feasible if and only if (s_α, Γ) is feasible.

In the rest of the appendix we will prove the main lemma. To illustrate the claim in the lemma, we express s_α and Λ by a directed graph that has labels on the vertices and edges, we call this graph a *sequence graph*. For example, when $s_\alpha = s_1$ and $\Lambda = [s_4 s_3 s_1 s_2, s_1 s_3 s_3, s_2 s_1 s_4, s_2 s_1]$, we have the directed graph in Fig. 4.

Let V denote the vertex set, then

$$V = \{s_0, s_1, s_2, \ldots, s_n\}$$

and the edge set is

$$E = \{(s_i, \Lambda_i[k])\} \bigcup \{(s_0, s_\alpha)\}$$

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

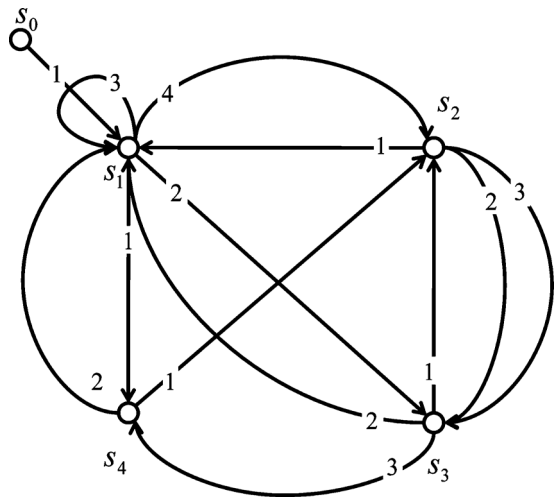

Fig. 4. An example of a sequence graph G.

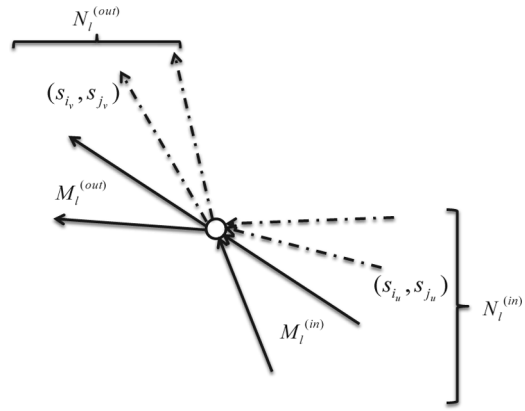

Fig. 5. An illustration of the incoming and outgoing edges of s_l. In which, the solid arrows indicate visited edges, and the dashed arrows indicate unvisited edges.

For each edge $(s_i, \Lambda_i[k])$, the label of this edge is k. For the edge (s_0, s_α), the label is 1. Namely, the label set of the outgoing edges of each state is $\{1, 2, \ldots\}$.

Given the labeling of the directed graph as defined above, we say that it contains a *complete walk* if there is a path in the graph that visits all the edges, without visiting an edge twice, in the following way: (1) Start from s_0. (2) At each vertex, we choose an unvisited edge with the minimal label to follow. Obviously, the labeling corresponding to (s_α, Λ) is a *complete walk* if and only if (s_α, Λ) is feasible. In this case, for short, we also say that (s_α, Λ) is a complete walk. Before continuing to prove the main lemma, we first give Lemma 15 and Lemma 16.

Lemma 15: Assume (s_α, Λ) with $\Lambda = [\Lambda_1, \Lambda_2, \ldots, \Lambda_\chi, \ldots, \Lambda_n]$ is a a complete walk, which ends at state s_χ. Then (s_α, Γ) with $\Gamma = [\Lambda_1, \ldots, \Gamma_\chi, \ldots, \Lambda_n]$ is also a complete walk ending at s_χ, if $\Lambda_\chi \equiv \Gamma_\chi$ (permutation).

Proof: (s_α, Λ) and (s_α, Γ) correspond to different labelings on the same directed graph G, denoted by L_1 and L_2. Since L_1 is a complete walk, it can travel all the edges in G one by one, denoted as

$$(s_{i_1}, s_{j_1}), (s_{i_2}, s_{j_2}), \ldots, (s_{i_N}, s_{j_N})$$

where $s_{i_1} = s_0$ and $s_{j_N} = s_\chi$. We call $\{1, 2, \ldots, N\}$ as the indexes of the edges.

Based on L_2, let's have a walk on G starting from s_0 until there is no unvisited outgoing edges to select. In this walk, assume the following edges have been visited:

$$(s_{i_{w_1}}, s_{j_{w_1}}), (s_{i_{w_2}}, s_{j_{w_2}}), \ldots, (s_{i_{w_M}}, s_{j_{w_M}})$$

where $w_1, w_2, \ldots, w_N$ are distinct indexes chosen from $\{1, 2, \ldots, N\}$ and $s_{i_{w_1}} = s_0$. In order to prove that L_2 is a complete walk, we need to show that: 1) $s_{j_{w_M}} = s_\chi$ and 2) $M = N$.

First, let's prove that $s_{j_{w_M}} = s_\chi$. In G, let $N_i^{(\text{out})}$ denote the number of outgoing edges of s_i and let $N_i^{(\text{in})}$ denote the number of incoming edges of s_i, then we have that

$$\begin{cases} N_0^{(\text{in})} = 0, N_0^{(\text{out})} = 1 \\ N_\chi^{(\text{in})} = N_\chi^{(\text{out})} + 1 \\ N_i^{(\text{in})} = N_i^{(\text{out})} \text{ for } i \neq 0, i \neq \chi \end{cases}$$

Based on these relations, we know that once we have a walk starting from s_0 in G, this walk will finally end at state s_χ. That is because we can always get out of s_i due to $N_i^{(\text{in})} = N_i^{(\text{out})}$ if $i \neq \chi, 0$.

Now, we prove that $M = N$. This can be proved by contradiction. Assume $M \neq N$, then we define

$$V = \{w_1, w_2, \ldots, w_M\}$$
$$\overline{V} = \{1, 2, \ldots, N\} / \{w_1, w_2, \ldots, w_M\}$$

where V corresponds to the visited edges based on L_2 and $\overline{V}$ corresponds to the unvisited edges based on L_2. Let $v = \min(\overline{V})$, then (s_{i_v}, s_{j_v}) is the unvisited edge with the minimal index. Let $l = i_v$, then (s_{i_v}, s_{j_v}) is an outgoing edge of s_l. Here $l \neq \chi$, because all the outgoing edges of s_χ have been visited. Assume the number of visited incoming edges of s_l is $M_l^{(\text{in})}$ and the number of visited outgoing edges of s_l is $M_l^{(\text{out})}$, then

$$M_l^{(\text{in})} = M_l^{(\text{out})}$$

see Fig. 5 as an example.

Note that the labels of the outgoing edges of s_l are the same for L_1 and L_2, since $l \neq \chi, 0$. Therefore, based on L_1, before visiting edge (s_{i_v}, s_{j_v}), there must be $M_l^{(\text{out})}$ outgoing edges of s_l have been visited. As a result, based on L_1, there must be $M_l^{(\text{out})} + 1 = M_l^{(\text{in})} + 1$ incoming edges of s_l have been visited before visiting (s_{i_v}, s_{j_v}). Among all these $M_l^{(\text{in})} + 1$ incoming edges, there exists at least one edge (s_{i_u}, s_{j_u}) such that $u \in \overline{V}$, since only $M_l^{(\text{in})}$ incoming edges of s_l have been visited based on L_2.

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

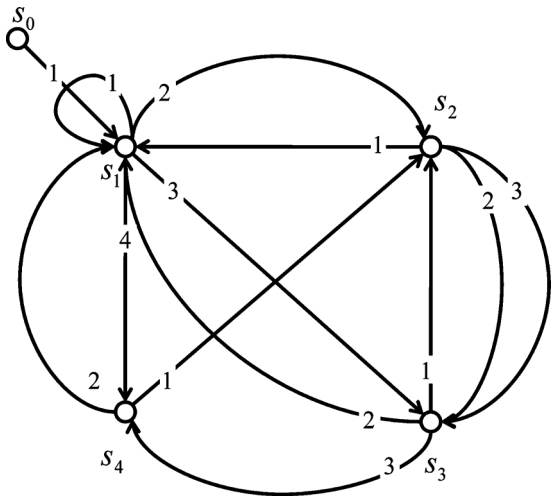

Fig. 6. The sequence graph G with new labels.

According to our assumption, both $u, v \in \overline{V}$, and v is the minimal one, so $u > v$. On the other hand, we know that (s_{i_u}, s_{j_u}) is visited before (s_{i_v}, s_{j_v}) based on L_1, so $u < v$. Here, the contradiction happens. Therefore, $M = N$.

This completes the proof. ■

Here, let's give an example of the lemma above. We know that, when $s_\alpha = s_1$, $\Lambda = [s_4s_3s_1s_2, s_1s_3s_3, s_2s_1s_4, s_2s_1]$, (s_α, Λ) is feasible. The labeling on a directed graph corresponding to (s_α, Λ) is given in Fig. 4, which is a complete walk starting at state s_0 and ending at state s_1. The path of the walk is

$$s_0s_1s_4s_2s_1s_3s_2s_3s_1s_1s_2s_3s_4s_1.$$

By permutating the labels of the outgoing edges of s_1, we can have the graph as shown in Fig. 6. The new labeling on G is also a complete walk ending at state s_1, and its path is

$$s_0s_1s_1s_2s_1s_3s_2s_3s_1s_4s_2s_3s_4s_1.$$

Based on Lemma 15, we have the following result.

Lemma 16: Given a starting state s_α and two collections of sequences $\Lambda = [\Lambda_1, \Lambda_2, \ldots, \Lambda_k, \ldots, \Lambda_n]$ and $\Gamma = [\Lambda_1, \ldots, \Gamma_k, \ldots, \Lambda_n]$ such that $\Gamma_k \doteq \Lambda_k$ (tail-fixed permutation). Then (s_α, Λ) and (s_α, Γ) have the same feasibility.

Proof: We prove that if (s_α, Λ) is feasible, then (s_α, Γ) is also feasible. If (s_α, Λ) is feasible, there exists a sequence X such that $s_\alpha = x_1$ and $\Lambda = \pi(X)$. Suppose its last element is $x_N = s_\chi$.

When $k = \chi$, according to Lemma 15, we know that (s_α, Γ) is feasible.

When $k \neq \chi$, we assume that $\Lambda_k = \pi_k(X) = x_{k_1}x_{k_2}\ldots x_{k_w}$. Let's consider the subsequence $\overline{X} = x_1x_2\ldots x_{k_w-1}$ of X. Then $\pi_k(\overline{X}) = \Lambda_k^{|\Lambda_k|-1}$ and the last element of $\overline{X}$ is s_k. According to Lemma 15, we can get that: there exists a sequence $x'_1x'_2\ldots x'_{k_w-1}$ with $x'_1 = x_1$ and $x'_{k_w-1} = x_{k_w-1}$ such that

$$\pi(x'_1x'_2\ldots x'_{k_w-1}) = [\pi_1(\overline{X}), \ldots, \Gamma_k^{|\Gamma_k|-1}, \pi_{k+1}(\overline{X}), \ldots, \pi_n(\overline{X})]$$

since $\Gamma_k^{|\Gamma_k|-1} \equiv \Lambda_k^{|\Lambda_k|-1}$.

Let $x'_{k_w}x'_{k_w+1}\ldots x'_N = x_{k_w}x_{k_w+1}\ldots x_N$, i.e., concatenating $x_{k_w}x_{k_w+1}\ldots x_N$ to the end of $x'_1x'_2\ldots x'_{k_w-1}$, we can generate a sequence $x'_1x'_2\ldots x'_N$ such that its exit sequence of state s_k is

$$\Gamma_k^{|\Gamma_k|-1} * x_{k_w} = \Gamma_k$$

and its exit sequence of state s_i with $i \neq k$ is $\Lambda_i = \pi_i(X)$.

So if (s_α, Λ) is feasible, then (s_α, Γ) is also feasible. Similarly, if (s_α, Γ) is feasible, then (s_α, Λ) is feasible. As a result, (s_α, Λ) and (s_α, Γ) have the same feasibility. ■

According to the lemma above, we know that $(s_\alpha, [\Lambda_1, \Lambda_2, \ldots, \Lambda_n])$ and $(s_\alpha, [\Gamma_1, \Lambda_2, \ldots, \Lambda_n])$ have the same feasibility, $(s_\alpha, [\Gamma_1, \Lambda_2, \ldots, \Lambda_n])$ and $(s_\alpha, [\Gamma_1, \Gamma_2, \ldots, \Lambda_n])$ have the same feasibility, $\ldots, (s_\alpha, [\Gamma_1, \Gamma_2, \ldots, \Gamma_{n-1}, \Lambda_n])$ and $(s_\alpha, [\Gamma_1, \Gamma_2, \ldots, \Gamma_{n-1}, \Gamma_n])$ have the same feasibility, so the statement in the main lemma is true.

REFERENCES

[1] A. V. Aho, L. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1976.

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

[3] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Second ed. New York: Wiley, 2006.

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

[5] T. S. Han and M. Hoshi, "Interval algorithm for random number generation," *IEEE Trans. Inf. Theory*, vol. 43, no. 2, pp. 599–611, 1997.

[6] T. S. Han, "Folklore in source coding: Information-spectrum approach," *IEEE Trans. Inf. Theory*, vol. 51, no. 2, pp. 747–753, 2005.

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

[8] D. Knuth and A. Yao, "The complexity of nonuniform random number generation," *Algorithms Complex.: New Directions and Recent Results*, pp. 357–428, 1976.

[9] J. von Neumann, "Various techniques used in connection with random digits," *Appl. Math. Ser., Notes by G. E. Forstyle, Nat. Bur. Stand.*, vol. 12, pp. 36–38, 1951.

[10] N. Nisan, "Extracting randomness: How and why. A survey," in *Proc. 11th Ann. IEEE Conf. Computat. Complex.*, 1996, pp. 44–58.

[11] S. Pae and M. C. Loui, "Optimal random number generation from a biased coin," in *Proc. 16th Ann. ACM-SIAM Symp. Discr. Algorithms*, 2005, pp. 1079–1088.

[12] Y. Peres, "Iterating von Neumann's procedure for extracting random bits," *Ann. Statist.*, vol. 20, pp. 590–597, 1992.

[13] B. Y. Ryabko and E. Matchikina, "Fast and efficient construction of an unbiased random sequence," *IEEE Trans. Inf. Theory*, vol. 46, pp. 1090–1093, 2000.

[14] P. A. Samuelson, "Constructing an unbiased random sequence," *J. Amer. Statist. Assoc*, pp. 1526–1527, 1968.

[15] R. Shaltiel, "Recent developments in explicit constructions of extractors," *BULLETIN-Eur. Assoc. Theoret. Comput. Sci.*, vol. 77, pp. 67–95, 2002.

[16] K. Visweswariah, S. R. Kulkarni, and S. Verdú, "Source codes as random number generators," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 462–471, 1998.

[17] Q. Stout and B. Warren, "Tree algorithms for unbiased coin tossing with a biased coin," *Ann. Probab.*, vol. 12, pp. 212–222, 1984.

[18] D. Zuckerman, "General weak random sources," in *Proc. 31st IEEE Symp. Found. Comput. Sci.*, 1990, pp. 534–543.

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.

Hongchao Zhou received the B.S. degree in physics and mathematics and the M.S. degree in control science and engineering from Tsinghua University, Beijing, China, in 2006 and 2008, respectively.

Currently, he is pursuing the Ph.D. degree in electrical engineering at California Institute of Technology, Pasadena. He worked with IBM China Research Laboratory, Beijing, as a research intern from January 2006 to June 2008. His current research interests include information theory, stochastic computation, and networks.

Jehoshua Bruck (S'86–M'89–SM'93–F'01) received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion–Israel Institute of Technology, Haifa, in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1989.

He is the Gordon and Betty Moore Professor of computation and neural systems and electrical engineering at the California Institute of Technology, Pasadena, CA. His extensive industrial experience includes working with IBM Almaden Research Center, San Jose, CA, as well as cofounding and serving as the Chairman of Rainfinity, acquired by EMC, Hopkinton, MA, in 2005; and XtremIO, San Jose, CA. His current research interests include information theory and systems and the theory of computation in biological networks.

Dr. Bruck is a recipient of the Feynman Prize for Excellence in Teaching, the Sloan Research Fellowship, the National Science Foundation Young Investigator Award, the IBM Outstanding Innovation Award, and the IBM Outstanding Technical Achievement Award. His papers were recognized in journals and conferences, including winning the 2010 IEEE Communications Society Best Student Paper Award in Signal Processing and Coding for Data Storage for his paper on codes for limited-magnitude errors in flash memories, the 2009 IEEE Communications Society Best Paper Award in Signal Processing and Coding for Data Storage for his paper on rank modulation for flash memories, the 2005 A. Schelkunoff Transactions Prize Paper Award from the IEEE Antennas and Propagation Society for his paper on signal propagation in wireless networks, and the 2003 Best Paper Award in the 2003 Design Automation Conference for his paper on cyclic combinational circuits.

Authorized licensed use limited to: SHANDONG UNIVERSITY. Downloaded on July 16,2022 at 04:30:28 UTC from IEEE Xplore. Restrictions apply.