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

I. Introduction

The problem of random number generation dates back to von Neumann [8] 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 [6], Elias [3], Stout and Warren [16] and Peres [11]. Elias [3] 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 [7] 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 [4] 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 [13]. 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 [3] 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 [1] 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 MC (Markov Chain). His algorithm approaches the maximum possible efficiency for a one-state MC. For a multi-state 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 [1] 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 can asymptotically reach the information-theoretic

Hongchao Zhou and Jehoshua Bruck are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125, USA, e-mail: hzhou@caltech.edu; bruck@caltech.edu.

This work was supported in part by the NSF Expeditions in Computing Program under grant CCF-0832824.

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 bits generation from finite Markov chains.

In this paper, we use the following notations:

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

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

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 $p_1, p_2, ..., p_n$ are unknown, the question is: How can we efficiently generate an independent and unbiased sequence of 0's and 1's from X? The efficiency (information-efficiency) of a generation algorithm is defined as the ratio between the expected length of the output sequence and the length of the input sequence, namely, the expected number of random bits generated per input symbol. In this section we describe three existing solutions for the problem of random bits generation from biased coins.

A. The von Neumann Scheme

In 1951, von Neumann [8] considered this question for biased coins and described a simple procedure for generating an independent unbiased binary sequence $z_1z_2...$ from the input sequence $X = x_1x_2....$ 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 $x_1x_2, x_3x_4, ...$ and use the following mapping for each pair

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

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) efficient, 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 [3] 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 with $1 \leq k \leq n$. 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:

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

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 $0 \leq c_k \leq 1$. The idea is that for any given class we partition the members of the class to groups of sizes that are a power of 2, for a group 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 output sequence should be ϕ (denotes the empty sequence). There are 4 sequences in S_1 and we assign the binary strings as follows:

$$s_1s_1s_1s_2 \rightarrow 00, \quad s_1s_1s_2s_1 \rightarrow 01$$

$$s_1s_2s_1s_1 \to 10, \quad s_2s_1s_1s_1 \to 11$$

Similarly, for S_2, there are 6 sequences that can be divided into a group of 4 and a group 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 $2^j \leq W < 2^{j+1}$. 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 [12], 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 with $n > 2$.

C. The Peres Scheme

In 1992, Peres [11] 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 $v \geq 2$ are defined inductively. Given input sequence $x_1x_2...x_{2m}$, let $i_1 < i_2 < ... < i_k$ denote all the indices $i \leq m$ for which $x_{2i} = x_{2i-1}$, then Ψ_v is defined as

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

Note that on the righthand side 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.

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

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

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 $\Psi : \{s_1, s_2, ..., s_n\}^N \to \{0,1\}^*$ 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.

That leads to the following property for Ψ. It says that given the number of s_i's for all i with $1 \leq i \leq n$, the number of such sequences to yield a binary sequence Y equals to that of sequences to yield Y' if 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 $1 \leq i \leq n$, 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 be a subset in $\{s_1, s_2, ..., s_n\}^N$ such that it includes all the sequences with the same number of s_i's for all i with $1 \leq i \leq n$, namely, $k_1, k_2, ..., k_n$. 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 $|S \bigcap B_Y| = |S \bigcap B_{Y'}|$.*

Proof: In S, the number of s_i's in each sequence is k_i for all $1 \leq i \leq n$, then we can get that

$$P[\Psi(X) = Y] = \sum_S |S \bigcap B_Y| \prod_{i=1}^n \beta(S)$$

where

$$\beta(S) = \prod_{i=1}^n p_i^{k_i}$$

Since $P[\Psi(X) = Y] = P[\Psi(X) = Y']$, we have

$$\sum_S (|S \bigcap B_Y| - |S \bigcap B_{Y'}|)\beta(S) = 0$$

The set of polynomials $\bigcup_S \{\beta(S)\}$ is linearly independent in the vector space of functions on $[0,1]$, so we can conclude that $|S \bigcap B_Y| = |S \bigcap B_{Y'}|$. ■

III. Some Properties of Markov Chains

Our goal is to efficiently generate random bits from a Markov chain with unknown transition probabilities. The paradigm 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 $X = x_1x_2...x_N$ with $x_i \in \{s_1, s_2, ..., s_n\}$, where $\{s_1, s_2, ..., s_n\}$ 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

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

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

of sequences $\pi(X) = [\pi_1(X), \pi_2(X), ..., \pi_n(X)]$, called exit sequences, where $\pi_i(X)$ is the sequence of states following s_i in X, namely,

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

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:

$$\begin{aligned}\pi_1(X) &= s_4s_3s_1s_2\\ \pi_2(X) &= s_1s_3s_3\\ \pi_3(X) &= s_2s_1s_4\\ \pi_4(X) &= s_2s_1\end{aligned}$$

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 [1], $x_1x_2...x_N$ 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 $x_1x_2...x_N$. ■

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

Proof: Note that the probability to generate X is

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

and the probability to generate Y is

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

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 $1 \leq i \leq n$. 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 [13], 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 [3], 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 $\pi_1(X), \pi_2(X), ...$ 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) $\Psi(\pi_1(X)) * \Psi(\pi_2(X)) * ...$ 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.

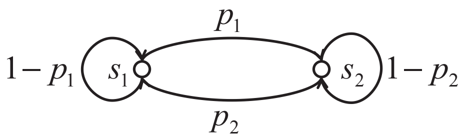

Fig. 1. An example of Markov chain with two states.

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 [1]: “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 a 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 non-zero probability we have

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

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

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

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 permutate 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 permutate 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 1 (Feasibility). *Given a Markov chain, a starting state s_α and a collection of sequences $\Lambda = [\Lambda_1, \Lambda_2, ..., \Lambda_n]$, 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 $\Lambda = [\Lambda_1, \Lambda_2, ..., \Lambda_n]$ and $\Gamma = [\Gamma_1, \Gamma_2, ..., \Gamma_n]$ 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.*

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

Lemma 5 (Feasible permutations of exit sequences). *Given an input sequence $X = x_1x_2...x_N$ with $x_N = s_\chi$ that produced from a Markov chain. Assume that $[\Lambda_1, \Lambda_2, ..., \Lambda_n]$ 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 $X' = x'_1x'_2...x'_N$ such that $x'_1 = x_1$ and $\pi(X') = [\Lambda_1, \Lambda_2, ..., \Lambda_n]$. 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 $X = x_1x_2...x_N$, 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 $x_1x_2...x_Ns_{n+1}$, whose exit sequences are

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

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

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

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 $x'_1x'_2...x'_Nx'_{N+1}$, we can get a new sequence $x = x'_1x'_2...x'_N$ such that its exit sequences are

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

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

$$X = s_1s_4s_2s_1s_3s_2s_3s_1s_1s_2s_3s_4s_1$$

with $\chi = 1$ and its exit sequences is 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 i^{th} 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

In the section above, we see that Elias suggested to paste the outputs of different exit sequences together, as the final output, but the simple direct concatenation cannot always work. By modifying the method of pasting these outputs, we get Algorithm A to generate unbiased random bits from any Markov chains.

Algorithm A
Input: A sequence $X = x_1x_2...x_N$ produced by a Markov chain, where $x_i \in S = \{s_1, s_2, ..., s_n\}$.
Output: A sequence 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, we still assume that an input sequence with length $N = 4$ is generated from this Markov chain and the starting state is 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 to be 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 to be 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 $\{s_1, s_2, ..., s_n\}^N$ into groups, and use G to denote the set of the groups. Two sequences X and X' are in the same group 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 $i \neq \chi$, $\pi_i(X') \doteq \pi_i(X)$.

We will show that for each group $S \in G$, the number of sequences to generate Y equals to that of sequences to generate Y' if Y and Y' have the same length, i.e., $|S \bigcap B_Y| = |S \bigcap B_{Y'}|$ if $|Y| = |Y'|$, where B_Y is the set of sequences of length N that yield Y.

Now, given a group S, if $i = \chi$ let's define S_i as the set 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 of all the permutations of $\pi_i(X)^{|\pi_i(X)|-1}$ for $X \in S$. According to Lemma 1, we know that for any $Y, Y' \in \{0,1\}^l$, there are the same number of members in S_i which generate Y and Y'. So we can use $|S_i(l)|$ to denote the number of members in S_i which generate a certain binary sequence with length l (e.g. Y).

According to the definitions above, let $l_1, l_2, ..., l_n$ be non-negative integers, then we have

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

where each combination $(l_1, l_2, ..., l_n)$ is a partition of the length of Y.

Similarly, we also have

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

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

Note that all the sequences in the same group S have the same probability to be generated. So when $|Y| = |Y'|$, the probability to generate 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 [3]. We can use the same argument in Elias's paper [3] to prove this theorem.

Let X_i denote the next state of s_i. Obviously, X_i is a random variable for $1 \leq i \leq n$, whose entropy 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

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

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 $1 \leq i \leq n$. 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 [3], 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 below bounded by

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

Then we have

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

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 $i \neq \chi$), 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 $(a, a, |\Lambda_i| - 2a)$ ... 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 $X = x_1x_2...$, 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 $x_1x_2...x_i$ and $x_ix_{i+1}...$ (note that both of the sequences have the element x_i). For the second sequence $x_ix_{i+1}...$, we can repeat the some procedure ... Iteratively, we can split a sequence X into several sequences such that they are independent of each other. These sequences, with the exception 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 [1], Blum proposed a beautiful algorithm to generate an independent unbiased sequence of 0's and 1's from any Markov chain by extending von Neumann scheme. His algorithm can deal with infinitely long sequences and use 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 scheme. In this section, we generalize Blum's algorithm by replacing von Neumann 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) $x_1x_2...$ produced by a Markov chain, where $x_i \in \{s_1, s_2, ..., s_n\}$.
Parameter: n positive integer functions (window size) $\varpi_i(k)$ with $k \geq 1$ for $1 \leq i \leq n$.
Output: A sequence (or a stream) 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 the input sequence is

$$X = s_1 s_1 s_1 s_2 s_2 s_2 s_1 s_2 s_2$$

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

$$E_1 = s_1 s_1 s_2 s_2 \quad E_2 = s_2 s_2 s_1$$

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 (9^{th}) symbol s_2, we get

$$E_1 = s_1 s_1 s_2 s_2 \quad E_2 = s_2 s_2 s_1 s_2$$

Since the current state of the Markov chain is s_2 and $|E_2| \geq 4$, we produce $\Psi(E_2 = s_2 s_2 s_1 s_2)$ and reset E_2 as ϕ.

In the example above, treating X as input to Algorithm B, we can get the output sequence is $\Psi(s_2 s_2 s_1 s_2)$. The algorithm does not output $\Psi(E_1 = s_1 s_1 s_2 s_2)$ 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 $k \in \{1, 2, ...\}$. 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 scheme) for Ψ. Assume a sequence of symbols $X = x_1 x_2 ... x_N$ 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

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

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.

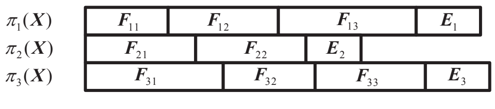

Fig. 2. The simplified expressions for the exit sequences of X.

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 $\{s_1, s_2, ..., s_n\}^N$ into groups, and use G to denote the group set. Two sequences X and X' are in the same group if and only if

1) $x_1 = x'_1$ and $x_N = x'_N$.
2) For all i with $1 \leq i \leq n$,
$$\pi_i(X) = F_{i1} F_{i2} ... F_{im_i} E_i$$
$$\pi_i(X') = F'_{i1} F'_{i2} ... F'_{im_i} E'_i$$
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, $E_i = E'_i$.

We will show that in each group $S \in G$, the number of sequences to generate Y equals to that of sequences to generate Y' if $|Y| = |Y'|$, i.e., $|S \bigcap B_Y| = |S \bigcap B_{Y'}|$ for $|Y| = |Y'|$, where B_Y is the set of sequences of length N that yield Y.

Now, given a group S, let's define S_{ij} as the set of all the permutations of F_{ij} for $X \in S$. According to Lemma 1, we know that for different $Y \in \{0,1\}^l$, there are the same number of members in S_{ij} which generate Y. So we can use $|S_{ij}(l)|$ to denote the number of members in S_{ij} which generate a certain binary sequence with length l.

Let $l_{11}, l_{12}, ..., l_{1m_1}, l_{21} ..., l_{nm_n}$ be non-negative integers such that their sum is $|Y|$, we want to prove that

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

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 $w + 1$.

Given an $X \in S$, assume F_{im_i} is the last segment that generates an output. According to our main lemma (Lemma 4), we know that for any sequence in S, F_{im_i} is always the last segment that generates an output. Now, let's fix 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

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

where $l_{11}, ..., l_{i(m_i - 1)}, l_{(i+1)1} ..., l_{nm_n}$ are non-negative integers. For a different l_{im_i}, there are $|S_{im_i}(l_{im_i})|$ choices for F_{im_i}. Therefore, $|S \bigcap B_Y|$ can be obtained by multiplying

$|S_{im_i}(l_{im_i})|$ 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 $1 \leq i \leq n$ 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 $U = (u_1, u_2, ..., u_n)$ 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 $(p_{i1}, p_{i2}, ..., p_{in})$.

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 $1 \leq i \leq n$.

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

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

With probability $1 - \epsilon_N$, we have

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

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

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

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 [10], 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 $n \times n$ non-negative integer matrix with $\sum_{i=1}^{n} \sum_{j=1}^{n} k_{ij} = N - 1$. We define $S_{(\alpha,K)}$ as*

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

where $k_j(X)$ is the number of s_j in X. A function $f : \{s_1, s_2, ..., s_n\}^N \to \{0, 1\}^*$ 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 $B_Y = \{X | X \in \{s_1, s_2, ..., s_n\}^N, f(X) = Y\}$ is the set of sequences of length N that yield Y.

Proof: 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, we can write

$$P[f(X) = Y] = \sum_{\alpha,K} |S_{(\alpha,K)} \bigcap B_Y| \phi(K) P(x_1 = s_\alpha)$$

where $\phi(K) = \prod_{i=1}^{n} \prod_{j=1}^{n} p_{ij}^{k_{ij}}$ and $\phi(K)P(x_1 = s_\alpha)$ is the probability to generate a sequence with starting state s_α and with exit sequences specified by K if such input sequence exists. Similarly,

$$P[f(X) = Y'] = \sum_{\alpha,K} |S_{(\alpha,K)} \bigcap B_{Y'}| \phi(K) P(x_1 = s_\alpha)$$

As a result,

$$\sum_{(\alpha,K)} (|S_{(\alpha,K)} \bigcap B_{Y'}| - |S_{(\alpha,K)} \bigcap B_Y|) \phi(K) P(x_1 = s_\alpha) = 0$$

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

$$\sum_{K} (|S_{(\alpha,K)} \bigcap B_{Y'}| - |S_{(\alpha,K)} \bigcap B_Y|) \phi(K) = 0$$

According to the linear independence of $\bigcup_K \{\phi(K)\}$ in the vector space of functions on $[0, 1]$, we can conclude that $|S_{(\alpha,K)} \bigcap B_Y| = |S_{(\alpha,K)} \bigcap B_{Y'}|$ for all (α, K) if $|Y| = |Y'|$.

Inversely, if for all Y, Y' with the same length, $|S_{(\alpha,K)} \bigcap B_Y| = |S_{(\alpha,K)} \bigcap B_{Y'}|$ for all (α, K), then Y and Y' have the same probability to be generated. Therefore, f can generate random bits from an arbitrary Markov chain. ■

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 Lemma 2.9 in [10], we know that

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

Then the average output length of h is

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

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

Algorithm C
Input: A sequence $X = x_1 x_2 ..., x_N$ produced by a Markov chain, where $x_i \in S = \{s_1, s_2, ..., s_n\}$.
Output: A sequence 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.

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 $n_1 > n_2 > ...$ 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 $\sum_{k=1}^{i} 2^{n_k} \leq r(x) < \sum_{k=1}^{i+1} 2^{n_k}$, 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 $1 \leq i \leq n$. 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

$$X = s_1 s_4 s_2 s_1 s_3 s_2 s_3 s_1 s_1 s_2 s_3 s_4 s_1$$

Its exit sequences is

$$\pi(X) = [s_4 s_3 s_1 s_2, s_1 s_3 s_3, s_2 s_1 s_4, s_2 s_1]$$

Then for this input sequence X, we have that

$$\theta(X) = [s_2, s_3, s_4, s_1]$$

$$\sigma(X) = [s_4 s_2 s_1, s_1 s_3, s_2 s_1, s_2]$$

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 $r(X)$, 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 [12].

Lemma 12. *$|S(X)|$ in Algorithm C is computable in $O(N(\log N \log\log N)^2)$ 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 $1 \leq i, j \leq n$. $\theta(X)$ 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 $\theta = (\theta_1, \theta_2, ..., \theta_n)$, 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 $\Lambda = [\Lambda_1, \Lambda_2, ..., \Lambda_n]$ by moving the first θ_i in $\pi_i(X)$ to the end for all $1 \leq i \leq n$. 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

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

$$=(\prod_{i=1}^{n}\frac{(k_{i1}+k_{i2}+...+k_{in})!}{k_{i1}!k_{i2}!...k_{in}!})(\prod_{i=1}^{n}\frac{k_{i\theta_i}}{(k_{i1}+k_{i2}+...+k_{in})})$$

where the first term, denoted by Z, is computable in $O(N(\log N\log\log N)^2)$ time [2]. Further more, we can get that

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

is also computable in $O(N(\log N\log\log N)^2)$ time. ■

Lemma 13. *$r(X)$ 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, $\sum_{\theta<\theta(X)}|S(X,\theta)|$ can be efficiently obtained by computing

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

where Z is defined in the last lemma. So far, we only need to compute $r(X,\theta(X))$, with respect to the lexicography order of $\sigma(X)$. $\sigma(X)$ can be written as a group of sequences $[\sigma_1(X),\sigma_2(X),...,\sigma_n(X)]$ such that for all $1\leq i\leq n$

$$\sigma_i(X)=\pi_i(X)^{|\pi_i(X)|-1}$$

There are $M=(N-1)-n$ symbols in $\sigma(X)$. Let $r_i(X)$ be the number of sequences $X'\in S(X,\theta(X))$ such that the first $M-i$ symbols of $\sigma(X')$ are the same with that of $\sigma(X)$ and the $M-i+1^{th}$ symbol of $\sigma(X')$ is smaller than that of $\sigma(X)$, then we can get that

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

Assume the $M-i+1^{th}$ symbol in $\sigma(X)$ is the u_i^{th} symbol in $\sigma_{v_i}(X)$. Then we can get that

$$r_i(X)=\sum_{s_w<\sigma_{v_i}[u_i]}\frac{k_w(T_i)}{|T_i|}\frac{|T_i|!}{k_1(T_i)!...k_n(T_i)!}\prod_{j>v_i}N_j(X)$$

where T_i is the subsequence of $\sigma_{v_i}(X)$ from the u_i^{th} symbol to the end; $N_j(X)$ is the number of permutations for $\sigma_j(X)$.

Let's define the values

$$\rho_i^0=\frac{|T_i|}{k_{w_i}(T_i)},\quad\lambda_i^0=\sum_{s_w<\sigma_{v_i}[u_i]}\frac{k_w(T_i)}{|T_i|}$$

where w_i is the index of the first symbol of T_i, i.e., $\sigma_{v_i}[u_i]=s_{w_i}$. Then $r(X,\theta(X))$ can be written as

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

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

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

$$s=1,2,...,\log M;i=1,2,...,2^{-s}M$$

Then applying the method in [12], we have that

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

which is computable in $O(M\log^3 M\log\log M)$ time. As a result, for a fixed 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.

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 a 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).

In the second calculation, we want to test the influence of window size ϖ (assume $\varpi_i(k)=\varpi$ for $1\leq i\leq n$) 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 $\{\frac{1}{n},\frac{1}{n},...,\frac{1}{n}\}$. 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 $(0.7228,1.1342,1.5827)$.

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

TABLE II
THE PROBABILITY OF EACH POSSIBLE OUTPUT SEQUENCE AND THE EXPECTED OUTPUT LENGTH.

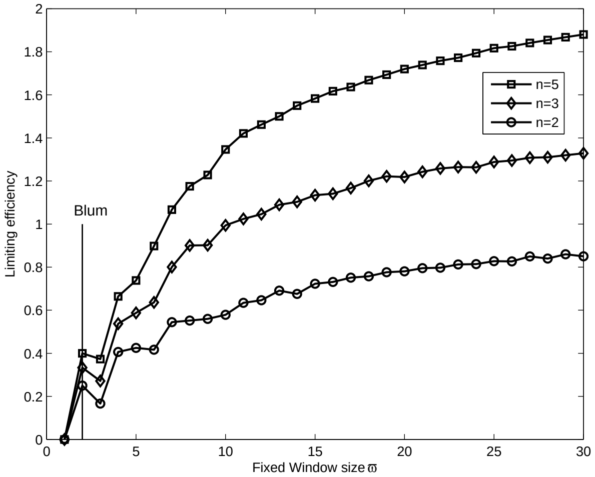

Fig. 3. The limiting efficiency of Algorithm B varies with the value of window size ϖ for different state number n, where we assume that the transition probability $p_{ij} = \frac{1}{n}$ for all $1 \leq i, j \leq n$.

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 $s_1 s_1 s_1$ or $s_2 s_2 s_2$, the Elias function will generate 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 to 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 source (introduced by Zuckerman [17]). Hence, the concept of an extractor was introduced - it converts weak random sequences to 'random-looking' sequences, using an additional small number of truly random bits. During the past two decades, extractors and their applications have been studied extensively, see [9] [14] 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-looking' sequences which are asymptotically close to random bits.

In information theory, it was discovered that optimal source codes can be used as universal random bits generators from arbitrary stationary ergodic random sources [15] [5]. 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 $\Lambda = [\Lambda_1, \Lambda_2, ..., \Lambda_n]$ and $\Gamma = [\Gamma_1, \Gamma_2, ..., \Gamma_n]$ 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

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

and the edge set is

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

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 $\{1, 2, ...\}$.

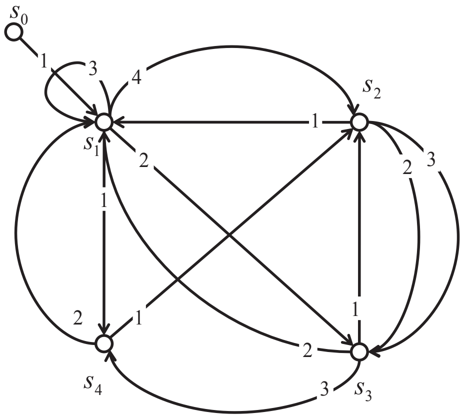

Fig. 4. An example of a sequence graph G.

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 14 and Lemma 15.

Lemma 14. *Assume (s_α, Λ) with $\Lambda = [\Lambda_1, \Lambda_2, ..., \Lambda_\chi, ..., \Lambda_n]$ is a a complete walk, which ends at state s_χ. Then (s_α, Γ) with $\Gamma = [\Lambda_1, ..., \Gamma_\chi, ..., \Lambda_n]$ 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

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

where $s_{i_1} = s_0$ and $s_{j_N} = s_\chi$. We call $\{1, 2, ..., N\}$ 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:

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

where $w_1, w_2, ..., w_N$ are distinct indexes chosen from $\{1, 2, ..., N\}$ 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 $N_i^{(out)}$ denote the number of outgoing edges of s_i and let $N_i^{(in)}$ denote the number of incoming edges of s_i, then we have that

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

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 $N_i^{(in)} = N_i^{(out)}$ if $i \neq \chi, 0$.

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

$$V = \{w_1, w_2, ..., w_M\}$$

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

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 $M_l^{(in)}$ and the number of visited outgoing edges of s_l is $M_l^{(out)}$, then

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

see Fig. 5 as an example.

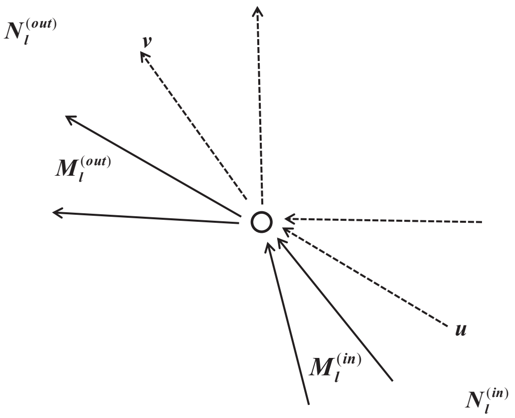

Fig. 5. An illustration of the incoming and outgoing edges of s_l. In which, the solid arrows indicate visited edges, and the dashed arrows indicate unvisited edges.

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

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

$$s_0s_1s_1s_2s_1s_3s_2s_3s_1s_4s_2s_3s_4s_1$$

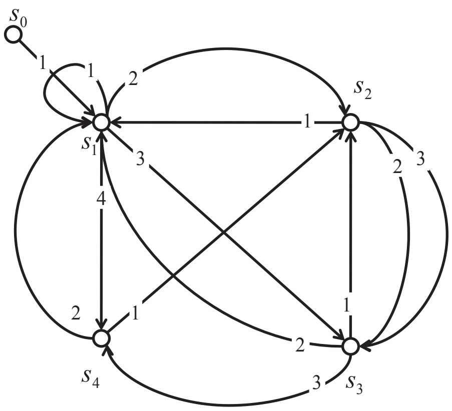

Fig. 6. The sequence graph G with new labels.

Based on Lemma 14, we have the following result

Lemma 15. *Given a starting state s_α and two collections of sequences $\Lambda = [\Lambda_1, \Lambda_2, ..., \Lambda_k, ..., \Lambda_n]$ and $\Gamma = [\Lambda_1, ..., \Gamma_k, ..., \Lambda_n]$ 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 14, we know that (s_α, Γ) is feasible.

When $k \neq \chi$, we assume that $\Lambda_k = \pi_k(X) = x_{k_1}x_{k_2}...x_{k_w}$. Let's consider the subsequence $\overline{X} = x_1x_2...x_{k_w-1}$ 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 14, we can get that: there exists a sequence $x_1'x_2'...x_{k_w-1}'$ with $x_1' = x_1$ and $x_{k_w-1}' = x_{k_w-1}$ such that

$$\pi(x_1'x_2'...x_{k_w-1}') = [\pi_1(\overline{X}), ..., \Gamma_k^{|\Gamma_k|-1}, \pi_{k+1}(\overline{X}), ..., \pi_n(\overline{X})]$$

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

Let $x_{k_w}'x_{k_w+1}'...x_N' = x_{k_w}x_{k_w+1}...x_N$, i.e., concatenating $x_{k_w}x_{k_w+1}...x_N$ to the end of $x_1'x_2'...x_{k_w-1}'$, we can generate a sequence $x_1'x_2'...x_N'$ 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 $(s_\alpha, [\Lambda_1, \Lambda_2, ..., \Lambda_n])$ and $(s_\alpha, [\Gamma_1, \Lambda_2, ..., \Lambda_n])$ have the same feasibility, $(s_\alpha, [\Gamma_1, \Lambda_2, ..., \Lambda_n])$ and $(s_\alpha, [\Gamma_1, \Gamma_2, ..., \Lambda_n])$ have the same feasibility, ..., $(s_\alpha, [\Gamma_1, \Gamma_2, ..., \Gamma_{n-1}, \Lambda_n])$ and $(s_\alpha, [\Gamma_1, \Gamma_2, ..., \Gamma_{n-1}, \Gamma_n])$ have the same feasibility, so the statement in the main lemma is true.

References

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

[2] P. B. Borwein, "On the complexity of calculating factorials", Journal of Algorithms, vol. 6, pp. 376-380, 1985.

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

[4] T. S. Han and M. Hoshi, "Interval algorithm for random number generation", IEEE Trans. on Information Theory, vol. 43, No. 2, pp. 599-611, 1997.

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

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

[7] D. Knuth and A. Yao, "The complexity of nonuniform random number generation", Algorithms and Complexity: New Directions and Recent Results, pp. 357-428, 1976.

[8] 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.

[9] N. Nisan, "Extracting randomness: how and why. A survey", in Proc. Eleventh Annual IEEE conference on Computational Complexity, pp. 44-58, 1996.

[10] S. Pae and M. C. Loui, "Optimal random number generation from a biased coin", in Proc. Sixteenth Annu. ACM-SIAM Symp. Discrete Algorithms, pp. 1079-1088, 2005.

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

[12] B. Y. Ryabko and E. Matchikina, "Fast and efficient construction of an unbiased random sequence", IEEE Trans. on Information Theory, vol. 46, pp. 1090-1093, 2000.

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

[14] R. Shaltiel, "Recent developments in explicit constructions of extractors", BULLETIN-European Association For Theoretical Computer Science, vol 77, pp. 67-95, 2002.

[15] K. Visweswariah, S. R. Kulkarni and S. Verdú, "Source codes as random number generators", IEEE Trans. on Information Theory, vol. 44, no. 2, pp. 462-471, 1998.

[16] Q. Stout and B. Warren, "Tree algorithms for unbiased coin tosssing with a biased coin", Ann. Probab., vol. 12, pp. 212-222, 1984.

[17] D. Zuckerman, "General weak random sources", in Proc. of the 31st IEEE Symposium on Foundations of Computer Science, pp. 534-543, 1990.