Efficiently Generating 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 Neumman [1] 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; 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 to generate random bits from a biased coin were proposed by Hoeffding and Simons [2], Stout and Warren [3] and Peres [4]. Elias [5] gave an optimal procedure such that the expected number of unbiased random bits generated per coin toss is asymptotically equal to the entropy of the biased coin. On the other hand, Knuth and Yao [6] gave a simple procedure to generate arbitrary distribution from an unbiased coin. Han and Hoshi [7] generalized this problem to consider the case that the given coin is biased with a known distribution.

In this paper, we study the problem of generating random bits from an arbitrary and unknown finite Markov chain. 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 [8]. 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 random source; obviously, this method is not efficient. Elias [5] suggested to make good use of all the states, for each state we produce an independent output sequence from the transitions out of this state, then the output sequence can be generated by pasting (concatenating) the collection of output sequences. However, neither Samuelson nor Elias proved that their methods work for arbitrary Markov chains. In fact, Blum [9] 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 Neumman 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 [9] worked on this problem and derived a beautiful algorithm to generate random bits from a degree-2 Markov chain *in expected linear time* by extending the single coin von Neumann scheme. 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 optimality due to the limitation (compared to Elias's algorithm) of von Neumman scheme.

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 arbitrary finite Markov chains, operates in expected linear time and achieves the information-theoretic upper bound on efficiency. Specifically, we propose Algorithm A, which is a simple modification of Elias's suggestion to generate random bits, it operates on finite sequences and its efficiency can reach the information-theoretic upper bound in the limit of long input sequences. There are several variants of Algorithm A. In addition, we propose Algorithm B, it is a combination of Blum's and Elias's algorithms, it generates infinitely long sequences of random bits in expected linear time. One of the ideas in our algorithms is that for the sake of achieving independence we ignore some input symbols. Hence, a natural question is: Can we improve the efficiency

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.

by minimizing the number of input symbols we ignore? We provide a positive answer to this question and describe Algorithm C, it is the first known optimal algorithm (in terms of efficiency) for random bits generation from an input sequence of length N, and it has polynomial time complexity.

The remainder of this paper is organized as follows. Section II introduces Elias's schemes to generate random bits from any biased coin. Section III presents our main lemma that allows the generalization to arbitrary Markov chains. Algorithm A is presented and analyzed in Section IV, it is a simple modification of Elias's suggestion for Markov chain. Algorithm B is presented in Section V, it is a generalization of Blum's algorithm. An optimal algorithm, called Algorithm C, is provided in Section VI, it can generate random bits from an arbitrary Markov chain with the maximal efficiency. Section VII addresses the computational complexity of our algorithms and shows that both Algorithm A and Algorithm C can generate outputs in $O(N \log^3 N \log\log N)$ time. Finally, Section VIII provides evaluation of our algorithms through simulations.

II. Elias's Scheme Generation of Random Bits

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

$$X = x_1 x_2 ... x_N \in \{s_1, s_2, ..., s_n\}^*$$

such that the probability to get s_i is p_i, with $\sum_{i=1}^{n} p_i = 1$. While we are given a sequence X the probabilities $p_1, p_2, ..., p_n$ are unknown, the question is: How can we generate an independent and unbiased sequence of 0's and 1's from X? The 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.

Elias [5] proposed an optimal (in terms of efficiency) generation algorithm; for the sake of completeness we describe it here. His 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 $1 \le k \le n$. 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:

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

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 $0 \le c_k \le 1$. 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 using 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:

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

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

Similarly, for S_2, there are 6 sequences that can be divided into a 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 m 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 odd number of members, there will be one and only one member assigned to ϕ (the empty string).

The foregoing assignment algorithm is not efficient as it is using a lookup table, however, efficient assignment can be achieved by using the lexicographic order. Given input sequence X of length N, the output sequence can be written as a function of X, denoted by $f_E(X)$, called Elias's function. Pae and Loui [10] showed that Elias's function is computable in polynomial time. Next we describe the procedure to compute $f_E(X)$ using the concept of a rank.

Computing the Elias Function

1) Given X, determine $|S(X)|$, the size of the class that includes X. It is the corresponding multinomial coefficient.
2) Compute the rank $r(X)$ of X in the class with respect to the lexicographical order (assume that the rank of the first element is 0).
3) Determine the output sequence based on $r(X)$ and $|S(X)|$, as follows: Let

$$|S(X)| = \alpha_l 2^l + \alpha_{l-1} 2^{l-1} + ... + \alpha_0 2^0$$

where $\alpha_l, \alpha_{l-1}, ..., \alpha_0$ is the binary expansion of integer $|S_k|$ with $\alpha_l = 1$. If $r(X) < 2^l$ then $f_E(X)$ is the l digit binary representation of X. If

$$\alpha_l 2^l + ... + \alpha_{i+1} 2^{i+1} \le r(X) < \alpha_l 2^l + ... + \alpha_i 2^i$$

then $f_E(X)$ is i digit binary representation of

$$r(X) - \alpha_l 2^l - \alpha_{l-1} 2^{l-1} - ... - \alpha_{i+1} 2^{i+1}$$

For example, consider the class S_2 in the example above, for each $X \in S_2$, we have

$$|S(X)| = 110$$

then we have the following:

$$r(s_1s_1s_2s_2) = 0\mathbf{00} \quad f_E(s_1s_1s_2s_2) = 00$$

$$r(s_1s_2s_1s_2) = 0\mathbf{01} \quad f_E(s_1s_2s_1s_2) = 01$$

$$r(s_1s_2s_2s_1) = 0\mathbf{10} \quad f_E(s_1s_1s_2s_2) = 10$$

$$r(s_2s_1s_1s_2) = 0\mathbf{11} \quad f_E(s_1s_1s_2s_2) = 11$$

$$r(s_2s_1s_2s_1) = 10\mathbf{0} \quad f_E(s_1s_1s_2s_2) = 00$$

$$r(s_2s_2s_1s_1) = 10\mathbf{1} \quad f_E(s_1s_1s_2s_2) = 1$$

The following property follows directly from the algorithm for computing the Elias function. It basically says that for a given class, if we generate a binary sequence of length k, we also generate all the possible 2^k binary sequences of length k. We will apply this property in the proofs in this paper.

Lemma 1 (Property of f_E)**.** *Given an input sequence X of length N such that $Y = f_E(X) \in \{0,1\}^k$. Then for any $Y' \in \{0,1\}^k$, there exists one and only one input sequence X', that is a permutation of X, such that $f_E(X') = Y'$.*

III. Main Lemma: Equivalence of Exit Sequences

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. We first define the following notations:

x_a	:	the a^{th} element of X
$X[a]$	:	the a^{th} element of X
$X[a:b]$	:	subsequence of X from the a^{th} to b^{th} element
X^a	:	$X[1:a]$
$Y \equiv X$	:	Y is a permutation of X
$Y \doteq X$	:	Y is a permutation of X and $y_{\|Y\|} = x_{\|X\|}$

The key idea is that for a given Markov chain, we can treat each state as a coin and treat its exits as the corresponding coin tosses. Namely, we can generate a collection 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,

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

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:

$$X = s_1\mathbf{s}_4s_2s_1\mathbf{s}_3s_2s_3s_1\mathbf{s}_1\mathbf{s}_2s_3s_4s_1$$

In general, 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 [9], $x_1x_2...x_N$ 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 $1 \le i \le n$.*

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 $1 \le i \le n$. 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 [8], Samuelson considered a two-state Markov chain, and he pointed out that it may generate unbiased random bits by applying von Neumman scheme to the exit sequence of state s_1. Later, in [5], in order to increase the efficiency, Elias has suggested a way to use all the symbols produced by a Markov chain. His main idea is to concatenate the output sequences that correspond to $\pi_1(X), \pi_2(X), ...$ as the final output. However, neither Samuelson nor Elias showed how to correctly paste output sequences of different parts to generate the final output, and they also did not prove that their methods can generate unbiased random bits for an arbitrary Markov chain. Now we consider two straightforward methods: (1) Only use $f_E(\pi_1(X))$ as the final output. (2) Concatenate all the sequences $f_E(\pi_1(X)), f_E(\pi_2(X)), ...$ as the final output. However, neither of them can generate random bits from an arbitrary Markov chain. Let's consider a simple example of a 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 an input

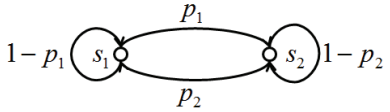


Fig. 1. An instance of Markov chain.

Input sequence	Probability	$f_E(\pi_1(X))$	$f_E(\pi_1(X))\|f_E(\pi_2(X))$
$s_1s_1s_1s_1$	$(1-p_1)^3$	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

sequence with 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 (based on direct concatenation) are given in Table I. We can see that when the input sequence length $N = 4$, the probabilities to produce a bit 0 or 1 are different for some p_1, p_2 in both of the methods.

It is not easy to figure out how to generate random bits from an arbitrary Markov chain, as Blum said in [9]: "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". What are the difficulties? The difficulties of generating random bits from an arbitrary Markov chain in expected linear (or polynomial) time come from the fact that it is hard to extract independence from the given Markov chain. It seems that the exit sequence of each state is independent since each exit of this state will not affect the other exits. However, this is not always true. When the length of the input sequence is given, say N, then the exit sequence of a state may not be independent. 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

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

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

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

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, not the length of X.

So far, we know that it is hard to get an independent sequence from an arbitrary Markov chain efficiently. So we have to consider this problem from another aspect. According to Lemma 3, we know that permutating the exit sequences does not change the probability of a sequence if the new sequence exists after the permutations. However, whether the new sequence exists or not depends on the permutations on the exit sequences. Let's go back to the initial example, where

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

and

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

If we permutate the last exit sequence $s_2 s_1$ to $s_1 s_2$, we cannot get a new sequence such that whose starting state is s_1 and exit sequences are

$$[s_4 s_3 s_1 s_2, s_1 s_3 s_3, s_2 s_1 s_4, s_1 s_2]$$

This can be verified by simply trying to construct a sequence, using the method of Blum (which is given in the proof of Lemma 2). But if we permutate the first exit sequence $s_4 s_3 s_1 s_2$ into $s_1 s_2 s_3 s_4$, we can find such a new sequence, which is

$$Y = s_1 s_1 s_2 s_1 s_3 s_2 s_3 s_1 s_4 s_2 s_3 s_4 s_1$$

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 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 and a 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 definition from the beginning of the section, 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: 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 $1 \le i \le n$. Then (s_α, Λ) is feasible if and only if (s_α, Γ) is feasible.*

Proof: In the rest of this section we will prove the main lemma. To illustrate the claim in the lemma, we express express s_α and Λ by a a directed graph that has labels on the vertices and edges. 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. 2. The vertex set

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

and the edge set

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

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

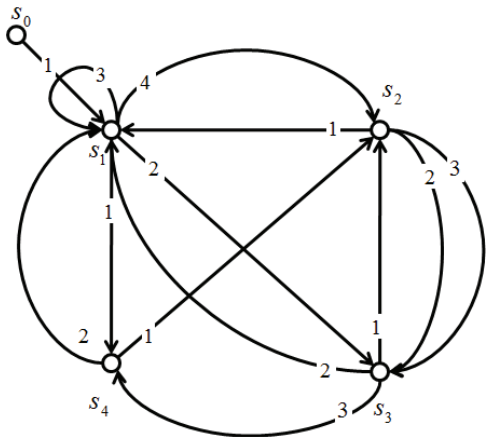


Fig. 2. Directed graph G with labels.

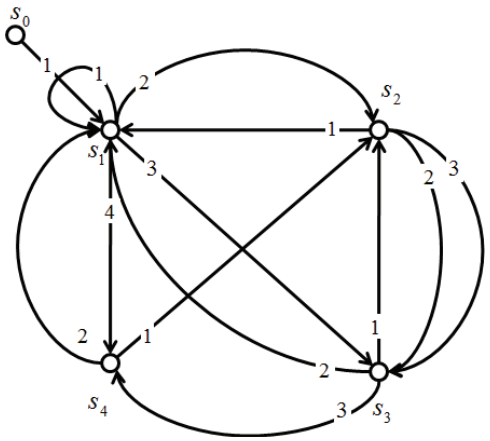


Fig. 3. Directed graph G with new labels.

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 and follow this edge. 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 5 and Lemma 6.

Lemma 5. *Assume* (s_α, Λ) *with* $\Lambda = [\Lambda_1, \Lambda_2, ..., \Lambda_n]$ *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: The proof of this lemma is given in the appendix. ■

For example, we know that, when $s_\alpha = s_1, \Lambda = [s_4 s_3 s_1 s_2, s_1 s_3 s_3, s_2 s_1 s_4, s_2 s_1]$, (s_α, Λ) is feasible. The labeling on a directed graph corresponding to (s_α, Λ) is given in Fig. 2, which is a complete walk starting at state s_0 and ending at state s_1. The path of the walk is

$$s_0 s_1 s_4 s_2 s_1 s_3 s_2 s_3 s_1 s_1 s_2 s_3 s_4 s_1$$

By permutating the labels of the outgoing edges of s_1, we can have the graph as shown in Fig. 3. The new labeling on G is also a complete walk ending at state s_1, and its path is

$$s_0 s_1 s_1 s_2 s_1 s_3 s_2 s_3 s_1 s_4 s_2 s_3 s_4 s_1$$

Based on Lemma 5 above, we have

Lemma 6. *Given a starting state* s_α *and two collections of sequences* $\Lambda = [\Lambda_1, \Lambda_2, ..., \Lambda_n]$ *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 5, 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 $\overline{X} = x_1 x_2 ... x_{k_w - 1}$ 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 5, 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

$$\pi(x'_1 x'_2 ... x'_{k_w-1}) = [\pi_1(\overline{X}), ..., \Gamma_k^{|\Gamma_k|-1}, \pi_{k+1}(\overline{X}), ..., \pi_n(\overline{X})]$$

because

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

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 $(s_\alpha, [\Gamma_1, \Gamma_2, ..., \Gamma_n])$ have the same feasibility, so the statement in the main lemma is true.

Here, we give an equivalent statement of the Main Lemma (Lemma 4).

Lemma 7. *Given an input sequence* $X = x_1 x_2 ... x_N$ *with* $x_N = s_\chi$*, produced from a Markov chain, for any* $[\Lambda_1, \Lambda_2, ..., \Lambda_n]$ *with*

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 one sequence $X' = x'_1 x'_2 ... x'_N$ *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 7 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 7 is true, then the statement in the Main Lemma is also true. Now we show that if the statement in the Main Lemma is true then the statement in Lemma 7 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 $x'_1 = x_1$. And more, we have $x'_N = s_\chi$.

This completes the proof. ■

We can demonstrate our results on the equivalence of the main lemma 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

$$X' = s_1s_1s_2s_3s_1s_3s_2s_1s_4s_2s_3s_4s_1$$

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 to paste these outputs, we get Algorithm A to generate unbiased random bits from any Markov chain.

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_χ.
for i := 1 to n do
  if i = χ then
    Output f_E(π_i(X)).
  else
    Output f_E(π_i(X)^{|π_i(X)|-1})
  end if
end for
```
Comment: $f_E(X)$ is Elias's function.

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 two simple methods (including direct concatenation) do 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 can be generated. In this case, the output sequence is independent and unbiased.

Now, our goal is to prove that all the sequences generated by Algorithm A are independent and unbiased. In order to prove this, we need to show that for any sequences Y and Y' of the same length, they have the same probability to be generated. Let $S_u(Y)$ denote the set of input sequences that underlies output sequence Y in Algorithm A. Then in the following lemma, we will show that there is a one-to-one mapping between $S_u(Y)$ and $S_u(Y')$ if $|Y| = |Y'|$.

Lemma 8 (Mapping for Algorithm A)**.** *In Algorithm A, assume input sequence $X = x_1x_2...x_N$ underlies an output sequence $Y \in \{0,1\}^k$ for some k, then for any $Y' \in \{0,1\}^k$, there exists one and only one sequence $X' = x'_1x'_2...x'_N$ which underlies Y' such that*

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

$$|f_E(\pi_i(X'))| = |f_E(\pi_i(X))|$$

3) *For all $i \neq \chi$, $\pi_i(X') \doteq \pi_i(X)$ and*

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

Proof: First we prove that for any Y' with $|Y'| = |Y|$, there exists such input sequence $X' = x'_1x'_2...x'_N$ satisfying all the requirements. Now, given a desired sequence Y', we

try to construct X'. Let's split Y' into several segments, $Y'_1, Y'_2, ..., Y'_n$, namely

$$Y' = Y'_1 Y'_2 ... Y'_n$$

such that the length of Y'_i for $1 \leq i \leq n$ is given by

$$|Y'_i| = \begin{cases} |f_E(\pi_i(X))| & \text{if } i = \chi \\ |f_E(\pi_i(X)^{|\pi_i(X)|-1})| & \text{if } i \neq \chi \end{cases}$$

According to Lemma 1, if $i = \chi$, we can find $\Lambda_i \equiv \pi_i(X)$ such that $f_E(\Lambda_i) = Y'_i$. If $i \neq \chi$, we can find Λ_i such that $\Lambda_i^{|\Lambda_i|-1} \equiv \pi_i(X)^{|\pi_i(X)|-1}$, $\Lambda_i[|\Lambda_i|] = \pi_i(X)[|\pi_i(X)|]$ and $f_E(\Lambda_i^{|\Lambda_i|-1}) = Y'_i$. Based on Lemma 7 (the equivalent statement of the Main Lemma), there exists a sequence $X' = x'_1 x'_2 ... x'_N$ with $x'_1 = x_1, x'_N = x_N$ and its exit sequences are $(\Lambda'_1, \Lambda'_2, ..., \Lambda'_n)$. It is not hard to check that X' satisfies all the requirements in the lemma.

Next, we prove that there are at most one sequence satisfying the requirements in the lemma. Assume there are two (or more) sequences U, V satisfying the requirements. According to Lemma 2, we know that they have different exit sequences. Then there exists a state s_i, such that $\pi_i(U) \neq \pi_i(V)$. If $i = \chi$, we also have $\pi_i(U) \equiv \pi_i(V)$, so we have $f_E(\pi_i(U)) \neq f_E(\pi_i(V))$, that means different Y'_i are produced, which contradicts with our assumption. If $i \neq \chi$, we will have $\pi_i(U) \doteq \pi_i(V)$, so we have $f_E(\pi_i(U)^{|\pi_i(U)|-1}) \neq f_E(\pi_i(V)^{|\pi_i(V)|-1})$, which also lead to contradictions.

This completes the proof. ∎

Theorem 1 (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: Assume the length of input sequence is N, then we want to show that for any $Y, Y' \in \{0,1\}^k$, Y and Y' have the same probability to be generated.

Let $S_u(Y)$ and $S_u(Y')$ denote the sets of input sequences underlying output sequences Y and Y'. According to Lemma 8 above, there is a one-to-one mapping between the elements in $S_u(Y)$ and $S_u(Y')$. Based on Lemma 3, we know that this mapping does not change the probability for sequences to be generated. As a result, from a Markov chain, the probability to generate a sequence in $S_u(Y)$ is equal to the probability to generate a sequence in $S_u(Y')$. So Y and Y' have the same probability to be generated. It means that any sequence of the same length k will be generated with the same probability. The result in the theorem is immediate from this conclusion. ∎

Theorem 2 (Efficiency)**.** *Let X be a sequence of length N generated by a Markov chain, which is used as input to algorithm A. 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 [5]. We can use the same argument in Elias's paper [5] 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 denote the stationary distribution of MC, 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 $f_E(.)$ when the input is $\pi_i(X)$ or $\pi_i(X)^{|\pi_i(X)|-1}$. According to Theorem 2 in Elias's paper [5], 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 below by

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

Then we have

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

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

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

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 sequences and does not lend itself to stream processing. In order to address these limitations, we can improve Algorithm A into two variants.

In the first variant of Algorithm A, instead of applying Elias's function 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 $k_{i1}, k_{i2}, ...$ then apply Elias's function 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 $k_{i1}, k_{i2}, ...$ 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 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 cut the sequence X. 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.

V. Algorithm B : Generalization of Blum's Algorithm

In [9], 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 Eliass algorithm) of von Neumman scheme. In this section, we propose Algorithm B by applying Elias scheme instead of von Neumann scheme, as a generalization of Blum's algorithm. Algorithm B keeps some good properties of Blum's algorithm, at the same time, its efficiency can get approach to 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:

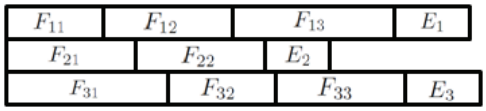
$E_i = \phi$ (empty) for all $1 \leq i \leq n$.
$k_i = 1$ for all $1 \leq i \leq n$.
c : the index of current state, namely, $s_c = x_1$.
while next input symbol is s_j ($\neq$ **null**) **do**
  if $|E_j| \geq \varpi_j(k_j)$ **then**
    Output $f_E(E_j)$.
    $E_j = \phi$.
    $k_j = k_j + 1$.
  end if
  $E_c = E_c s_j$ (Add s_j to E_c).
  $c = j$.
end while

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

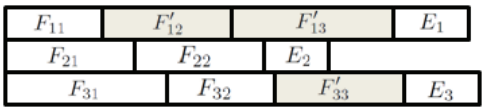
$$X = s_1s_1s_1s_2s_2s_2s_1s_2s_2$$

The exit sequences of X is

$$\pi(X) = [s_1s_1s_2s_2, s_2s_2s_1s_2]$$



(a) X



(b) $\overline{X}$

Fig. 4. The simplified expressions for the exit sequences of X and $\overline{X}$.

If we treat X as input to the algorithm above, we can get the output sequence is $f_E(s_2s_2s_1s_2)$. Here, the algorithm does not output $f_E(s_1s_2s_2s_2)$ 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 $k \in \{1, 2, ...\}$. Namely, Algorithm B is a generalization of Blum's algorithm. Assume a sequence of symbols $X = x_1x_2...x_N$ 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. 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. 4(a) for simple illustration.

Lemma 9 (Mapping for Algorithm B)**.** *In Algorithm B, assume input sequence $X = x_1x_2...x_N$ underlies an output sequence $Y \in \{0,1\}^k$ for some k, then for any $Y' \in \{0,1\}^k$, there exists one and only one sequence $X' = x'_1x'_2...x'_N$ which underlies Y' such that*

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

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

$$\pi_i(X') = F'_{i1}F'_{i2}...F'_{im_i}E_i$$

3) *For all i, j, we have*

$$F_{ij} \equiv F'_{ij} \text{ and } |f_E(F_{ij})| = |f_E(F'_{ij})|$$

Proof: Let's construct such a sequence X' satisfying all the requirements in the lemma:

1) Let $S = \{(i,j)|\forall 1 \leq i \leq n, 1 \leq j \leq m_i\}$. Let $w = \sum_i m_i$ be the number of segments generated from X.

2) Construct a sequence $\overline{X}$ with

$$\pi_i(\overline{X}) = \overline{F}_{i1}\overline{F}_{i2}...\overline{F}_{im_i}E_i$$

such that

a) $\overline{x}_1 = x_1$.

b) For all i, j,

$$\overline{F}_{ij} = \begin{cases} F_{ij} & \text{if } (i,j) \in S \\ F'_{ij} & \text{otherwise} \end{cases}$$

as shown in Fig. 4(b). According to the equivalent statement of the main lemma (Lemma 7), we know that such sequence $\overline{X}$ exists and it is unique.

3) Let $\overline{X}$ be the input sequence, and $\overline{F}_{i_1j_1}, \overline{F}_{i_2j_2}, ..., \overline{F}_{i_wj_w}$ be the ordered segments used to produce outputs, i.e., when the input sequence is $\overline{X}$, the output sequence is

$$f_E(\overline{F}_{i_1j_1})f_E(\overline{F}_{i_2j_2})...f_E(\overline{F}_{i_wj_w})$$

4) Let $k = \max\{m | 1 \le m \le w, (i_m, j_m) \in S\}$, then we do the following operations:

a) Let $S = S/(i_k, j_k)$.

b) Let $l = |f_E(F_{i_kj_k})|$, then set $F'_{i_kj_k}$ such that

$$F'_{i_kj_k} \equiv F_{i_kj_k}$$

and

$$f_E(F'_{i_kj_k}) = Y'[|Y'| - l + 1 : |Y'|]$$

c) Let $Y' = Y'[1 : |Y'| - l]$.

5) Repeat step 2-4) until S becomes empty (ϕ).

6) Repeat step 2) and 3) and let $X' = \overline{X}$.

We want to prove that the X' constructed using the procedure above satisfies all the requirements in Lemma 6. It is obvious that X' satisfies the conditions 1) to 3) in Lemma 6, the only thing that we need to prove is that X' underlies Y'.

Note that we have to repeat step 2) and 3) for $w+1$ times. We want to prove that after the k^{th} iteration of step 3), the following conclusion is true:

1) Let $G = f_E(\overline{F}_{i_{w-k+2}j_{w-k+2}})...f_E(\overline{F}_{i_wj_w})$, which is a suffix of Y'.

2) $S = \{(i_m, j_m) | 1 \le m \le w - k + 1\}$

Let's prove the conclusion above by induction. First, after the first iteration, we can get that $G = \phi$, so the conclusion is true. Now, let's assume that this conclusion is true after the k^{th} iteration, we will prove that this conclusion is also true after the $k+1^{th}$ iteration when $k \le w$. In the following proof, all the symbols are for the k^{th} iteration. In order to distinguish with the k^{th} iteration, we use $*$ to mark the symbols for the $k+1^{th}$ iteration.

According to our assumption, when input sequence is $\overline{X}$, the output is

$$f_E(\overline{F}_{i_1j_1})f_E(\overline{F}_{i_2j_2})...f_E(\overline{F}_{i_wj_w})$$

where $\{(i_m, j_m) | 1 \le m \le w - k + 1\} = S$ and G is a suffix of Y'. According to step 4), we know that $\overline{X}^*$ is constructed from X by replacing $F_{i_{w-k+1}j_{w-k+1}}$ with $F'_{i_{w-k+1}j_{w-k+1}}$. In the following, we prove that $\overline{X}^*$ satisfies the properties in the conclusion.

Let $\overline{x}_\alpha$ be the α^{th} symbol in $\overline{X}$ such that once $\overline{x}_\alpha$ is provided the algorithm outputs $f_E(F_{i_{w-k-1}j_{w-k-1}})$. When the input sequence is $\overline{X}^\alpha$, which is a subsequence of $\overline{X}$, the output sequence will be

$$f_E(F_{i_1j_1})f_E(F_{i_2j_2})...f_E(F_{i_{w-k-1}j_{w-k-1}})$$

According to the equivalent statement of the main lemma (Lemma 7), we know that if $F_{i_{w-k+1}j_{w-k+1}}$ is replaced with $F'_{i_{w-k+1}j_{w-k+1}}$, we can get a new sequence U^α with length α such that when the input sequence is U^α, the algorithm will output $f_E(F_{i_1j_1}), ..., f_E(F_{i_{w-k}j_{w-k}})$ in some order and output $f_E(F'_{i_{w-k+1}j_{w-k+1}})$ at the end (after reading the last symbol). Note that, after processing U^α, all the values of the variables stored are the same as those after processing $\overline{X}^\alpha$. If we concatenate U^α with $\overline{X}[\alpha + 1 : N]$ we can get a new sequence $\overline{X}^* = U^\alpha\overline{X}[\alpha + 1 : N]$, which is exactly the $\overline{X}^*$ mentioned above.

For this sequence $\overline{X}^*$, it has the following properties:

1) Its output sequence ends with

$$f_E(F'_{i_{w-k+1}j_{w-k+1}})\overline{F}_{i_{w-k+2}j_{w-k+2}})...f_E(\overline{F}_{i_wj_w})$$

2) $S^* = S/(i_{w-k+1}, j_{w-k+1})$.

So far, it is not hard to see that

$$f_E(F'_{i_{w-k+1}j_{w-k+1}})\overline{F}_{i_{w-k+2}j_{w-k+2}})...f_E(\overline{F}_{i_wj_w})$$

is a suffix of Y' and

$$(i^*_m, j^*_m) = (i_m, j_m) \quad \text{for all } w - k + 1 \le m \le w$$

As a result, the conclusion above is true for the $k + 1^{th}$ iteration. By induction, we know that the conclusion is true for all $1 \le k \le w + 1$. Let $k = w + 1$, we can get that

$$f_E(\overline{F}_{i_1j_1}) + ... + f_E(\overline{F}_{i_wj_w}) = Y'$$

i.e., $X' = \overline{X}$ underlies Y' and this X' satisfies all the requirements in the lemma.

Now, we show that there are at most one such sequence X' satisfying all the requirements in the lemma. Assume there are two sequences satisfying all the requirements, such that their outputs are

$$f_E(G_{i_1j_1})f_E(G_{i_2j_2})...f_E(G_{i_wj_w})$$

and

$$f_E(H_{i'_1j'_1})f_E(H_{i'_2j'_2})...f_E(H_{i'_wj'_w})$$

where G_{ij} and H_{ij} are segments to produce outputs for the two input sequences.

By induction, we try to prove that for any $1 \le k \le w$, $i_k = i'_k, j_k = j'_k$ and $G_{i_kj_k} = H_{i'_kj'_k}$. When $k = w$, according to Lemma 7, it is not hard to get $i_w = i'_w, j_w = j'_w$. Therefore, $G_{i_wj_w} \equiv H_{i'_wj'_w}$. We also know that $f_E(G_{i_wj_w}) = f_E(H_{i'_wj'_w})$, so $G_{i_kj_k} = H_{i'_kj'_k}$. Finally, the conclusion is true for $k = w$. Assume the conclusion is true for all $l < k \le w$, using the same argument as above, we can also get that the conclusion is true for l. As a result, the conclusion is true for all $1 \le k \le w$. Based on Lemma 2 for uniqueness, we can say that the two sequences are the same, i.e, the sequence X' is unique.

This completes the proof. ∎

Theorem 3 (Algorithm B)**.** *Let the sequence generated by a Markov chain be used as input to algorithm B, then Algorithm B can generate an independent unbiased sequence in expected linear time.*

Proof: In order to prove the theorem above, we can use the same method of the proof for Algorithm A. The only difference is that in the proof we use the mapping obtained in Lemma 9. ∎

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 important to analyze the relationship between scheme efficiency and window size ϖ.

Theorem 4 (Efficiency)**.** *Let MC be a Markov chain with transition matrix P. Let the sequence generated by MC be 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 Elias's scheme when the input sequence of length ϖ is generated by a n-face coin with distribution $(p_{i1}, p_{i2}, ..., p_{in})$.

Proof: Assume the input sequence is X. 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. By analyzing Elias's scheme, we can get

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

Based on this formula, we can study the relationship between the limiting efficiency and the window size numerically (see Section VIII). When the window size becomes large, the limiting efficiency ($n \to \infty$) can get approach to the information-theoretic upper bound.

VI. Algorithm C : Optimal Algorithm

In this section, we try to construct an optimal algorithm, such that its information-efficiency is maximized, no matter how big the length of the input sequence is. Following the definition of Pae and Loui in [10], we define randomizing functions:

Definition 2 (Randomizing function [10])**.** *A function $f : \{s_1, s_2, ..., s_n\}^N \to \{0,1\}^*$ is randomizing function if for each k and for each $w \in \{0,1\}^{k-1}$,*

$$P[Y[k] = 0 | Y[1, k-1] = w] = P[Y[k] = 1 | Y[1, k-1] = w]$$

for any Markov chain, where $Y \in \{0,1\}^*$ is the output of the function.

According to the definition, an algorithm can generate independent unbiased sequences if and only if it can be treated as a randomizing function.

Lemma 10 (Necessary condition for Randomizing function)**.** *If a function $f : \{s_1, s_2, ..., s_n\}^N \to \{0,1\}^*$ is a randomizing function, then for each $w \in \{0,1\}^*$, $|B_{w0}| = |B_{w1}|$, where*

$$B_w = \{X \in \{s_1, s_2, ..., s_n\}^N | f(X)^{|w|} = w\}$$

Let $K = \{k_{ij}\}$ be an $n \times n$ non-negative integer matrix with $\sum_{i=1}^{n} \sum_{j=1}^{n} k_{ij} = N-1$. Let's define S_K as

$$S_K = \{X \in \{s_1, s_2, ..., s_n\}^N | k_j(\pi_i(X)) = k_{ij}\}$$

where $k_j(X)$ is the function to count the number of s_j in X. Then for any such matrix K, we have

$$|B_{w0} \bigcap S_K| = |B_{w1} \bigcap S_K|$$

Proof: Pae and Loui proved this theorem in the case for coin flips. Using the same argument, we can prove the case for Markov chain. According to Lemma 2.2 in [10]: A function f is randomizing if and only if $P[X \in B_{w0}] = P[X \in B_{w1}]$.

Here we can write

$$P[X \in B_{w0}] = \sum_K |B_{w0} \bigcap S_K| \phi(K)$$

where $\phi(K) = \prod_{i=1}^{n} \prod_{j=1}^{n} p_{ij}^{k_{ij}}$ is the probability to generate a sequence with exit sequences specified by K if such input sequence exists. It is easy to get that

$$\phi(K) = \prod_{i=1}^{n} \prod_{j=1}^{n} p_{ij}^{k_{ij}}$$

with p_{ij} as the transition probability from state i to state j, and $\sum_{i,j=1}^{n} k_{ij} = N-1$

Similarly,

$$P[X \in B_{w1}] = \sum_K |B_{w1} \bigcap S_K| \phi(K)$$

So we have

$$\sum_K (|B_{w0} \bigcap S_K| - |B_{w1} \bigcap S_K|) \phi(K) = 0$$

The set of polynomials $\bigcup_K \{\phi(K)\}$ is linearly independent in the vector space of functions on $[0,1]$. We can conclude that

$|B_{w0} \bigcap S_K| = |B_{w1} \bigcap S_K|$. Since $|B_{w0}| = \sum_K |B_{w0} \bigcap S_K|$, we can also get $|B_{w0}| = |B_{w1}|$. ∎

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 following theorem.

Lemma 11 (Sufficient condition for optimal algorithm)**.** *If there exists a randomizing function f^* such that for 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,*

$$\sum_{X \in S_K} |f^*(X)| = \alpha(|S_K|)$$

then f^* is optimal in efficiency to generate independent unbiased sequence from a Markov chain with unknown transition probabilities.

Proof: Let h denote an arbitrary randomizing function. According to Lemma 2.9 in [10], we know that

$$\sum_{X \in S_K} |h(X)| \leq \alpha(|S_K|)$$

Then the average output length of h is

$$\frac{1}{N} \sum_K \sum_{X \in S_K} |h(X)| \phi(K) \leq \frac{1}{N} \sum_K \alpha |S_K| \phi(K)$$

$$= \frac{1}{N} \sum_K \sum_{X \in S_K} |f^*(X)| \phi(K)$$

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

Based on the lemmas above, we can construct the following algorithm (Algorithm C) which can generate unbiased random bits from an arbitrary Markov chain with maximal 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 0s and 1s.
Main Function:

1) Get the matrix $K = \{k_{ij}\}$ with

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

2) Define $S(X)$ as

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

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.

In Algorithm C, for an input sequence X with $x_N = s_\chi$, we can rank it with respect to the lexicography 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 lexicography order defined above, both $|S(X)|$ and $r(X)$ can be obtained using brute-force search. However, it needs too many computations. In the next section, we can show that both $|S(X)|$ and $r(X)$ is computable in almost expected linear time. In the rest of this section, we show that Algorithm C can generate independent unbiased sequences from any Markov chain.

Theorem 5 (Algorithm C)**.** *Let the sequence generated by a Markov chain be used as input to algorithm C, then the output of Algorithm C is an independent unbiased sequence. And more, Algorithm C is optimal.*

Proof: Assume the length of input sequence is N, then we want to show that for any $Y \in \{0,1\}^K$ and $Y' \in \{0,1\}^K$, Y and Y' have the same probability to be generated.

Let $S_u(Y)$ and $S_u(Y')$ denote the set of input sequences underlying output sequence Y and Y'. For each sequence $X \in S_u(Y)$, there exists one and only one K such that $X \in S_K$. According to Lemma 1, there exists one and only one sequence $X' \in S_K$ such that X' underlies Y'. Since both X and X' are in S_K, they have the same probability to be generated. So all the elements in $S_u(Y)$ and $S_u(Y')$ are one-to-one mapping. As a result, Y and Y' have the same probability to be generated. Using the same argument as Theorem 1, we know that Algorithm C can generate independent unbiased sequences.

Let's treat algorithm C as a function f_C, then f_C is a randomizing function. It is not hard to prove that f_C satisfies the condition in Lemma 11. So f_C is optimal, furthermore, Algorithm C is optimal. ∎

In Algorithm A, 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 $\frac{H(X)}{N}$.

VII. Fast Computation of Algorithms

In [11], Ryabko and Matchikina showed that when the input sequence with length N is generated by a biased coin with two faces, then Elias's function is computable in $O(N \log^3 N \log\log(N))$ time. This result is a little surprising, because $|S_k|$ has $O(N)$ bits and there are still a lot of operations on $|S_k|$ to compute Elias's function. In this section, we will first generalize Ryabko and Matchikina's procedure to the case that the input sequence with length N is generated by a biased n-face coin instead of a biased 2-face

coin. We will show that their conclusion about computational complexity is still true. As a result, Algorithm A is computable in $O(N\log^3 N\log\log(N))$ time. We also apply the same idea of fast computation to the optimal algorithm (Algorithm C) such that the optimal algorithm is also computable in $O(N\log^3 N\log\log(N))$ time.

A. Fast Computation of Elias's Function

Let's define the order on states: (1) $s_i < s_j$ iff $i < j$ (2) $s_i = s_j$ iff $i = j$. Given any sequence $X \in \{s_1, s_2, ..., s_n\}^*$, we let $k_i(X)$ be the number of s_i's in X for all $1 \le i \le n$. Then, we can write $k(X) = (k_1(X), k_2(X), ..., k_n(X))$ as a counting function.

Given an input sequence $X = x_1x_2...x_N$ with $x_i \in \{s_1, s_2, ..., s_n\}$ from a n-face coin, we can produce the output sequence $f_E(X)$ based on the procedure in section II. In the procedure, the class

$$S(X) = \{X'|k(X') = k(X)\}$$

and $r(X)$ is the rank of X in the class $S(X)$ with respect to the lexicographical order. During this procedure, the computation of $|S(X)|$ and $r(X)$ dominates the computational complexity of $f_E(X)$. In [12], P. Borwein pointed out that $N!$ is computable in $O(N(\log N\log\log N)^2)$ time. As a result, $|S(X)|$ is computable in $O(N(\log N\log\log N)^2)$ time. Now, let's consider the computation of $r(X)$. Following the idea of [11], we suppose $\log N$ to be an integer. Otherwise, we can add s_0's at the beginning of X to make $\log N$ an integer. It does not change the rank of X in $S(X)$.

Let's define $r_i(X)$ as the number of sequences $X' \in S(X)$ such that $X'^{N-i} = X^{N-i}$ and $x'_{N-i+1} < x_{N-i+1}$, then we can get that

$$r(X) = \sum_{i=1}^{N} r_i(X)$$

Let T_i denote the subsequence of X from the $(N-i+1)^{th}$ symbol to the end, then we have

$$r_i(X) = \sum_{s_w < x_{N-i+1}} \frac{k_w(T_i)}{i} \frac{i!}{k_1(\tau_i)!...k_n(\tau_i)!}$$

Define the values

$$\rho_i^0 = \frac{i}{k_{w_i}(T_i)}, \quad \lambda_i^0 = \sum_{s_w < x_{N-i}} \frac{k_w(T_i)}{i}$$

where w_i is the index of the state x_{N-i+1}, namely, $x_{N-i+1} = s_{w_i}$.

Then

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

In order to quickly calculate $r(X)$, the following calculations are performed:

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

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

It is not difficult to get that[1]

$$r(X) = \lambda_1^{\log N}$$

Using the same argument as [11], we know that $r(X) = \lambda_1^{\log N}$ is computable in $O(N\log^3 N\log\log N)$ time, which leads us to the following lemma.

Lemma 12 (Computational Complexity of f_E)**.** *Given any input sequence $X = x_1x_2...x_N$ with $x_i \in \{s_1, s_2, ..., s_n\}$ generated from a biased n-face coin, Elias's function $f_E(X)$ is computable in $O(N\log^3 N\log\log N)$ time.*

Based on this lemma, it is easy to get a corollary about Algorithm A.

Theorem 6 (Computational Complexity of Algorithm A)**.** *Let the sequence generated by a Markov chain be used as input to algorithm A. If the length of input sequence is N, then the output sequence can be obtained in $O(N\log^3 N\log\log N)$ time.*

B. Fast Computation of the Optimal Algorithm

Now, let's think about the computational complexity of Algorithm C, which is also dominated by $|S(X)|$ and $r(X)$. Intuitively, in order to get the optimal output sequence using Algorithm C, we may need much more computations than Algorithm A. Surprisingly, as the same as Algorithm A, we will show that in Algorithm C, $|S(X)|$ is computable in $O(N(\log N\log\log N)^2)$ time and $r(X)$ is computable in $O(N\log^3 N\log\log N)$ time.

Lemma 13. *$|S(X)|$ 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 $1 \le i, j \le n$. $\theta(X)$ is the vector of the last symbols of $\pi(X)$ defined as in the last section. 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 efficiently 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 $1 \le i \le n$. 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. According to the main lemma, we can do tail-fixed permutations on $\Lambda_1, \Lambda_2, ..., \Lambda_n$, and the yielded new sequences still belong to $S(X,\theta)$. So we can get

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

[1]Here, different from [11], we don't have the term $|S(X)|$, because we use different expressions for ρ_i^0 and λ_i^0.

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

where the first term, denoted by Z, is computable in $O(N(\log N \log\log N)^2)$ time. 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 14. *$r(X)$ 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 lexicography 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

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

in which $\sum_{\theta<\theta(X)} |S(X,\theta)|$ can be efficiently obtained by computing

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

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 $1 \le i \le n$

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

There are $M = (N-1) - n$ symbols in $\sigma(X)$. Let $r_i(X)$ be the number of sequences $X' \in S(X,\theta(X))$ 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 is the u_i^{th} symbol of $\sigma_{v_i}(X)$. 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 $\sigma_j(X)$.

We can see that if we 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, then applying the method in [11], 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, in Algorithm C, $|S(X)|$ is computable in $O(N(\log N \log\log N)^2)$ time and $r(X)$ is computable in $O(N \log^3 N \log\log N)$ time. So we can have the following theorem about the computational complexity of Algorithm C.

Theorem 7 (Computational Complexity of Algorithm C)**.** *Let the sequence generated by a Markov chain be used as input to algorithm C. If the length of input sequence is N, then the output sequence can be obtained in $O(N \log^3 N \log\log N)$ time.*

VIII. Experimental Results

In this section, we implement Algorithm A, Algorithm B, and Algorithm C on personal computer. Then we verify these algorithms and evaluate their performances.

In the first experiment, we randomly generate a transition matrix for a Markov chain with state number $n = 3$, which is

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

Assume a sequence with length 12 is generated from the Markov chain defined above, and the first state of this sequence is s_1. So there are $3^{11} = 177147$ possible input sequences. For each possible input sequence, we can compute its generating probability and corresponding output sequence under different algorithms. By enumerating all the possible input sequences, we can calculate the probabilities of all possible output sequences under different algorithms, as shown in Table II. We can see that all these algorithms can generate independent unbiased sequences for the randomly produced Markov chain. Relatively, Algorithm C has the highest efficiency, since it is optimal, and Algorithm A has higher efficiency than Algorithm B.

In the second experiment, we want to test the influence of window size ϖ (assume $\varpi_i(k) = \varpi$ for $1 \le i \le n$) to the efficiency of Algorithm B. However, the efficiency depends on the transition matrix of the Markov chain. In order to evaluate the effect of window size ϖ, we just assume that each entry of the transition matrix is $\frac{1}{n}$, where n is the state number, and the input sequence is infinitely long. In this case, the stationary distribution of the Markov chain is $\{\frac{1}{n},\frac{1}{n},...,\frac{1}{n}\}$. In Fig. 5, it shows that when $\varpi = 2$ (Blum's Algorithm), the limiting efficiencies for $n = (2,3,5)$ are $(\frac{1}{4},\frac{1}{3},\frac{2}{5})$ separately. 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.

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 UNDER DIFFERENT ALGORITHMS.

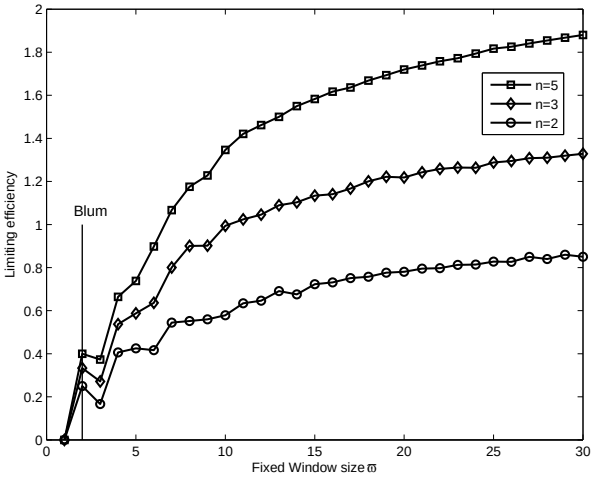


Fig. 5. The limiting efficiency of Algorithm B varies with the value of window size ϖ for different state number n, where we assume that the transition probability $p_{ij} = \frac{1}{n}$ for all $1 \leq i, j \leq n$.

When ϖ becomes larger, the efficiency of Algorithm B will converge to the information-theoretical upper bound $\log_2 n$. Note that 3 is not a good value for the window size in the algorithm, but it is not surprising: it is intuitively correct if we consider the case of $n = 2$.

IX. CONCLUSIONS

In this paper, we considered a traditional problem that how to generate independent unbiased sequences from any Markov chain in expected linear time or with the maximal efficiency. First, this is an important problem in computer science. One application is mentioned by Blum [9]: The bits produced by most of physical sources appear to be completely random, and the probability of each bit is conditional on the preceding bits. In this case, we can treat these bits as a Markov chain and apply the results in this paper, to produce the seeds required by pseudorandom number generators. Besides the applications in computer science, the results in the paper also can be applied in coding theory. In order to address this problem, 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. There are still some open questions, for example, if there is some noise on the Markov chain, how can we generate random bits efficiently such that they can be used in randomizing algorithms?

APPENDIX

Lemma 5. *Assume* (s_α, Λ) *with* $\Lambda = [\Lambda_1, \Lambda_2, ..., \Lambda_n]$ *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 prove that (1) $s_{j_{w_M}} = s_\chi$ and (2) $M = N$. Now, we prove them separately.

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 staring 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 $i \neq \chi$.

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. 6 as an example.

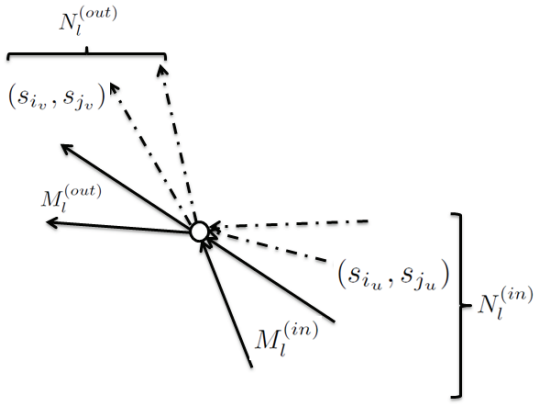


Fig. 6. An illustration of the incoming and outgoing edges of s_l. In which, the solid arrows indicate visited edges, and the dashed arrows indicate unvisited edges.

Note that the labels of the outgoing edges of s_l are the same for L_1 and L_2, since $l \neq \chi$. 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. ■

REFERENCES

[1] 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.

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

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

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

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

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

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

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

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

[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. 1079C1088, 2005.

[11] 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.

[12] P.B. Borwein, "On the complexity of calculating factorials", Jounal of Agorithms 6, pp. 376-380, 1985.