Random Permutations using Switching Networks

Artur Czumaj*
Department of Computer Science
Centre for Discrete Mathematics and its Applications (DIMAP)
University of Warwick
A.Czumaj@warwick.ac.uk

ABSTRACT

We consider the problem of designing a simple, oblivious scheme to generate (almost) random permutations. We use the concept of switching networks and show that almost every switching network of logarithmic depth can be used to almost randomly permute any set of $(1-\varepsilon)n$ elements with any $\varepsilon > 0$ (that is, gives an almost $(1-\varepsilon)n$-wise independent permutation). Furthermore, we show that the result still holds for every switching network of logarithmic depth that has some special expansion properties, leading to an explicit construction of such networks. Our result can be also extended to an explicit construction of a switching network of depth $\mathcal{O}(\log^2 n)$ and with $\mathcal{O}(n \log n)$ switches that almost randomly permutes any set of n elements. We also discuss basic applications of these results in cryptography.

Our results are obtained using a non-trivial coupling approach to study mixing times of Markov chains which allows us to reduce the problem to some random walk-like problem on expanders.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

Keywords

random permutations; Markov chains; switching networks

1. INTRODUCTION

The problem of efficiently generating random, almost random, or pseudo-random permutations is one of the central problems in complexity, distributed computing, and cryptography. In this paper, we consider the problem of oblivious generation of almost random permutations through mixing elements in layered networks. A *switching network* $\mathfrak{N}$ of depth $\mathfrak{d}$ is a layered network with $\mathfrak{d}+1$ layers, each layer having n nodes. The nodes between consecutive layers are connected by *switches* (cf. Figure 1). A switch between two input nodes at layer ℓ and two output nodes at layer $\ell+1$ takes the two inputs and either transposes them (if the switch is active) or leaves them unchanged (if the switch is inactive). The switches are disjoint, that is, if a switch takes nodes at positions i and j at layer ℓ and outputs them to positions i' and j' at layer $\ell+1$, then there is no other switch that connects any of the four nodes involved (has as its input node i or j, or has as its output node i' or j').

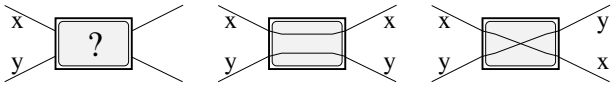

Figure 1: A *switch* and its two possible outcomes (depending on the random outcome of coin toss).

Switching networks have been extensively studied in the context of sorting networks, where each switch (called a comparator there) sorted the input elements (see, e.g. [1, 20, 21, 22]). While the role of a switch in sorting networks is to sort the numbers from the two incoming inputs, in our case, the switch performs a *uniformly random choice* of which incoming element in a switch from layer ℓ will go to which outgoing node at layer $\ell+1$. A special property of sorting networks is that they give oblivious sorting algorithms, which is a desirable feature in several applications. It is also known that switching networks can be used to generate almost random permutations, where the randomness is achieved by setting the switches at random. Since the input network is fixed, very simple, and the switches are oblivious (only their behavior is random), mixing properties of switching networks have attracted attention in various areas, most notably in cryptography (see e.g., [28] and [29] and in numerous follow-up papers) and in distributed systems (see, e.g., [7]).

One can view the process of mixing elements using switching networks in the framework of *card shuffling* (cf. [3]). A single shuffle puts some cards (input elements) into pairs and then swaps (and rearranges) the cards in each pair at random. A card shuffling process would repeatedly apply the shuffle using the different possible ways of pairing the cards. In this setting, the way in which the cards are paired would be determined by a network, and so, if in the ℓ^{th} shuffle, cards at positions i and j are paired, then we would have a switch between nodes at positions i and j. For exam-

*Research partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP), and by EPSRC awards EP/D063191/1 and EP/G064679/1.

STOC'15, June 14–17, 2015, Portland, Oregon, USA.

ACM 978-1-4503-3536-2/15/06 ...$15.00.
DOI: http://dx.doi.org/10.1145/2746539.2746629.

ple, the well-known Thorp shuffle (see, e.g., [3, 9, 23]) for n cards in each round takes cards from location i and $n/2+i$, $1 \le i \le n/2$, and moves them to locations $2i-1$ and $2i$, and their internal order is uniformly chosen independently of all other choices. If we apply this process repeatedly and if n is a power of 2, then we can model the Thorp shuffle using switching networks with the underlying networks being the butterfly networks (cf. Figure 2 and [3, 9, 23]).

1.1 Mixing using switching networks

To formally introduce switching networks that permute elements let us first introduce some basic notation. A *matching* of $\{1,\ldots,n\}$ is a set of pairs $\{i,j\} \subseteq \{1,\ldots,n\}$ with $i \neq j$ such that no element from $\{1,\ldots,n\}$ appears in more than one pair. A pair $\{i,j\}$ in a matching is called a *matching edge*. A *perfect matching* of $\{1,\ldots,n\}$ is a matching of $\{1,\ldots,n\}$ of size exactly $\frac{n}{2}$ (throughout the paper we assume that n is even, though see Remark 1). Let $\mathfrak{M}^{\mathfrak{d}}$ be the set of all sequences $(M_0,\ldots,M_{\mathfrak{d}-1})$ such that each M_t is a perfect matching of $\{1,\ldots,n\}$. For a given $\mathbf{M}^{\mathfrak{d}} = (M_0,\ldots,M_{\mathfrak{d}-1}) \in \mathfrak{M}^{\mathfrak{d}}$, we define a switching network $\mathfrak{N}$ of depth $\mathfrak{d}$ so that the switches between layers i and $i+1$ are determined by the edges (pairs) of matching M_i. See also Figure 2.

Every layered network $\mathfrak{N}$ corresponding to $\mathbf{M}^{\mathfrak{d}}$, $\mathbf{M}^{\mathfrak{d}} = (M_0,\ldots,M_{\mathfrak{d}-1}) \in \mathfrak{M}^{\mathfrak{d}}$, defines in a natural way a stochastic process (Markov chain) $(\mathcal{Q}_t)_{t=0}^{\mathfrak{d}}$ on state space of all permutation of $\{1,\ldots,n\}$ with the transition rules $\mathcal{Q}_t \mapsto \mathcal{Q}_{t+1}$ that for each matching edge $\{i,j\} \in M_t$ exchanges the items in nodes i and j independently at random with probability $\frac{1}{2}$. Throughout the paper, the process $(\mathcal{Q}_t)_{t=0}^{\mathfrak{d}}$ will be called the *random shuffling process* for network $\mathfrak{N}$ and every transformation $\mathcal{Q}_t \mapsto \mathcal{Q}_{t+1}$ will be called the *shuffle* defined by matching M_t. In other words, the random shuffling process for $\mathfrak{N}$ is the process that takes as the input n elements and run them through network $\mathfrak{N}$ with switches making random selections of the outputs.

We note that even though we would like to generate a uniformly random permutation, this is known to be impossible for the processes considered in this paper (since we cannot achieve the probability of $\frac{1}{n!}$ using only random switches, which can only generate probabilities of the form $k \cdot 2^{-s}$) and we will settle with *almost random* permutations.

1.2 Mixing of partial permutations

In this paper we will focus on the task of generating almost random *partial permutations*, that is, of mixing *almost* all input objects. A *k-partial n-permutation* is any sequence $\langle x_0,\ldots,x_{n-1}\rangle$ consisting of k 0s and $n-k$ distinct elements from $1,\ldots,n-k$. The set of all k-partial n-permutations is denoted by $\mathbb{S}_{n,k}$; $\mathbb{S}_{n,k} = \{(x_0,\ldots,x_{n-1}) : |\{j \in \{0,1,\ldots,n-1\} : x_j = 0\}| = k \text{ and } \forall_{r\in\{1,\ldots,n-k\}}\exists_{j\in\{0,1,\ldots,n-1\}}\ x_j = r\}$. Observe that $|\mathbb{S}_{n,k}| = \frac{n!}{k!}$. In this paper our focus will be on the case when k is arbitrarily small, except that $k = \Omega(n)$.

We consider the problem of generating an almost uniformly random element from $\mathbb{S}_{n,k}$: an almost random k-partial n-permutation. Notice that this task is equivalent to the problem of generating an *almost random $(n-k)$-wise independent permutation*, using the terminology frequently used in the complexity and cryptographical setting, see, e.g., [19, 28].

Our main technical result, Theorem 3.5, shows that almost every switching network (all but at most a $\frac{1}{n^2}$ fraction) of logarithmic depth almost randomly permutes any set of $(1-\varepsilon)n$ elements with any $\varepsilon > 0$ (equivalently, it almost uniformly generates random k-partial n-permutations for any $k = \Omega(n)$, or it generates almost $(1-\varepsilon)n$-wise independent permutations).

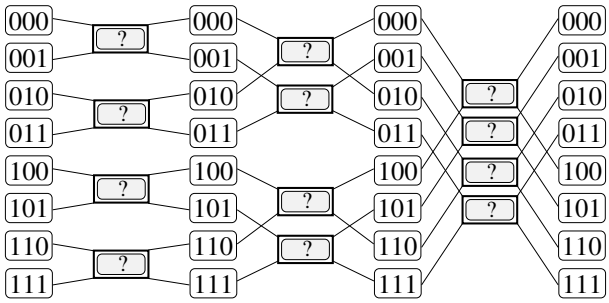

Figure 2: Switching network modeled by a single butterfly network (corresponding to three rounds of Thorp-shuffle) for $n = 8$. In this case, $\mathbf{M}^{\mathfrak{d}} = (M_0,\ldots,M_{\mathfrak{d}-1}) \in \mathfrak{M}^{\mathfrak{d}}$ with $\mathfrak{d} = 3$, and each M_ℓ, $0 \le \ell \le \mathfrak{d}-1$, consists of switches between nodes whose positions differ only in the $(\ell+1)^{\text{st}}$ rightmost bit (where we use binary representation for the positions of the nodes).

In fact, our approach is stronger and we can show that if the switching network $\mathfrak{N}$ has some specific expansion property (is $\mathfrak{good}$, as defined in Section 2.2) and if it has depth at least $c \log n$ for some large enough constant c, then it almost randomly permutes any set of $(1-\varepsilon)n$ elements (Theorem 3.4). Since one can construct networks that are $\mathfrak{good}$ using existing expander constructions, this gives an explicit switching network of logarithmic depth that almost randomly permutes any set of $(1-\varepsilon)n$ elements (Theorem 3.6).

REMARK 1. *While our analysis will assume that n is even, one could use essentially identical arguments for odd n, in which case instead of using perfect matchings of size $\frac{n}{2}$ to define $\mathfrak{N}$, we would use almost perfect matchings of size $\frac{n-1}{2}$. Furthermore, our result for almost all switching networks can be easily extended to the case when the matchings defining $\mathfrak{N}$ are not necessarily perfect, but instead, all are of size at least cn for any positive constant c.*

Furthermore, while we have defined the random shuffling process to use only perfect matchings, one can define the process for any sequence of matchings (if an element is unmatched between two layers ℓ and $\ell+1$, then the location from layer ℓ will be retained in layer $\ell+1$). In fact, our construction in Theorem 1.1 does not use perfect matchings.

1.3 Generating almost random permutations

We note that our construction can be easily applied to design a switching network of depth $\mathcal{O}(\log^2 n)$ that generates almost random permutations. We first apply a switching network $\mathfrak{N}$ of depth $\mathcal{O}(\log n)$ to partition n elements into two sets of size $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$, respectively, almost uniformly at random, and then we recursively apply the same network to each of the two sets. To ensure that the error is small in all recursive calls, for any set of elements (even if it is significantly smaller than n) we use a switching network of depth $\mathcal{O}(\log n)$ to make the partition (the depth of the network is independent of the actual size of the instance of the recur-

sive call). This gives a construction of a switching network of depth $\mathcal{O}(\log^2 n)$ almost randomly permuting n elements.

Furthermore, let us observe that this construction can be modified to ensure that the network will have only $\mathcal{O}(n \log n)$ switches (though still, having depth of $\mathcal{O}(\log^2 n)$). (This idea has been used earlier in a similar context by Morris and Rogaway [27].) Indeed, after applying a switching network $\mathfrak{N}$ of depth $\mathcal{O}(\log n)$ to partition n elements into two sets of size $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$, we note that the set of $\lceil \frac{n}{2} \rceil$ elements is almost randomly chosen and is almost randomly permuted, and as the result, one needs to apply the recursive calls only to the remaining $\lfloor \frac{n}{2} \rfloor$ elements. If, as before, for any set of elements (even if it is less than n) we use a switching network of depth $\mathcal{O}(\log n)$ to make the partition, then the total size of network satisfy the recurrence $S(m) = cm \log n + S(\lfloor m/2 \rfloor)$, and thus it has $\mathcal{O}(n \log n)$ switches.

THEOREM 1.1. *There is an explicit switching network of depth $\mathcal{O}(\log^2 n)$ and with $\mathcal{O}(n \log n)$ switches that almost randomly permutes any set of n elements.*

We will give a more formal statement of Theorem 1.1 with formal arguments in Section 3.2.1.

1.4 Techniques

Our analysis of switching networks uses a novel method of analyzing convergence times of Markov chains extending the approach presented recently in [9] to study the behavior of butterfly network. The approach uses non-Markovian couplings to reduce the problem to a certain combinatorial problem of existence of some basic structure in the switching networks. Informally, our result shows that if a switching network $\mathfrak{N}$ can be split into two parts, the first one having embedded some binary trees (called *fundamental trees* in Section 3.1.1) and the second one having some perfect matching on a subset of objects (called *fundamental matching* in Section 3.1.2), then $\mathfrak{N}$ almost randomly permutes any set of $(1-\varepsilon)n$ elements. The main challenge is then to show the existence of such structures in almost all switching networks and in all $\mathfrak{good}$ switching networks of logarithmic depth.

While some parts of our analysis closely follow the framework already proposed in [9], the need of dealing with an almost unknown network (unlike the explicitly given butterfly networks underlying Thorp shuffle, which are well defined and well understood) requires several new tools to be developed. In particular, one of the central challenges stems from the fact that in our paper we cannot assume that an individual element will reach a random position very quickly, and it requires $\mathcal{O}(\log n)$ layers to achieve this situation. A similar property holds also for butterfly networks, where one needs $\Theta(\log n)$ layers for an element to reach a random location, but this is a key obstacle why the known results (cf. [9, 26]) for such networks require a superlogarithmic depth to permute partial permutations (rather than $\mathcal{O}(\log n)$ depth, as for the switching networks considered in this paper).

1.5 Related works

In a sequence of papers, Morris [23, 24, 25] proved that switching networks defined by a polylogarithmic number of butterfly networks (corresponding to Thorp shuffle) can be used to generate almost random permutations. This gives a switching network of depth $\mathcal{O}(\log^3 n)$ (only for n being a power of two; for general n, the depth is $\mathcal{O}(\log^4 n)$) that generates almost random permutations. The existence of switching networks of polylogarithmic depth follows also from earlier works, e.g., [7, 29]. Our construction gives depth $\mathcal{O}(\log^2 n)$ for any n, and in fact, it gives networks with only $\mathcal{O}(n \log n)$ switches (though the depth is $\mathcal{O}(\log^2 n)$).

There have been also some work focusing on the analysis of random properties of shallow switching networks. Morris et al. [26] showed that after applying T times butterfly networks (which gives a switching network of depth $T \log n$), the network (almost) randomly permutes any set of $n^{1-1/T}$ elements; more concretely, the distance from the distribution of any set of q elements differs from the uniform distribution by at most $\frac{q}{T+1}(\frac{4q \log n}{n})^T$. Note that this approach gives a useful bound only for $q < \frac{n}{4 \log n}$. Czumaj and Vöcking [9] considered larger q and showed that for any $\varepsilon > 0$, any set of $(1-\varepsilon)n$ elements will be almost randomly permuted after applying $\mathcal{O}(\log n)$ times the butterfly networks (giving a switching network of depth $\mathcal{O}(\log^2 n)$).

Very recently, Gelman and Ta-Shma [13] studied the quality of generating partial permutations after applying a single Benes network of depth $2 \log n$. They show that a single Benes switching network permutes well any set of up to $\sqrt{n}$ elements (more precisely, for any q, the distance from the distribution of any set of q elements differs from the uniform distribution by at most $\frac{q(q-1)}{2n}$).

1.6 Cryptographic applications

Switching networks have been studied in the past to generate random objects, most notably because of their applications in cryptography. For example, motivated by applications to security of some cryptographic protocols, Rackoff and Simon [29, Theorem 3.1] show that almost every switching network of polylogarithmic depth almost randomly shuffles any sequence of $\frac{n}{2}$ 0s and $\frac{n}{2}$ 1s; this result has been further improved in [7, Theorem 2.3], who reduced the bound on the mixing time from polylogarithmic (with a two-digit degree [31]) to $\mathcal{O}(\log n)$. Our construction (Theorem 3.7) yields a similar result (and for an arbitrary number of 0s and 1s) but is also contructive, and can be directly applied in the context of cryptographic defense against traffic analysis (cf. Rackoff and Simon [29]).

The idea of using oblivious processes (card shuffling) to generate random or pseudo-random permutations has also been used in other areas of cryptography, most notably thanks to the ideas suggested by Moni Naor (cf. [19, 28]). In some applications, the key feature required is to be able to trace the trajectory of every single element in secure computations without needing to know the positions of too many of other elements, or seeing other computations. Clearly, this property trivially holds for switching networks, where one can trace the trajectory of any element by following its path in the switching network, and thus by checking only the outcome of $\mathfrak{d}$ switches in the network of depth $\mathfrak{d}$. In a sequence of papers [16, 26, 27, 30] there have been various "shuffling" algorithms that provide provably-secure block ciphers even for adversaries that can observe the encryption of all domain points. Morris et al. [26] have been using switching networks (Thorp shuffle, also called maximally unbalanced Feistel network in this setting) to achieve fully secure pseudorandom permutations secure for $n^{1-\varepsilon}$ queries in logarithmic number of rounds. This result has been improved recently, first to retain the security to $(1-\varepsilon)n$ queries with a logarithmic number rounds [16], then to n queries with $\mathcal{O}(\log^2 n)$ rounds [30], and finally to n queries and $\mathcal{O}(\log n)$ calls *on average*

to the one-bit-output random function [27]. However, unlike the original approach of Morris et al. [26], the improved schemes have not been using switching networks and rely on the *swap-or-not scheme*. The underlying operation (in our terminology) was to set up the switches such that for every layer ℓ, one chooses a random number $K_\ell \in \{0,1\}^{\log_2 n}$, and then set a switch between each element at position X (given in binary) and $K_\ell \oplus X$. Note that this does not define a switching network, since the choices of K_ℓ in each layer are not given in advance and correspond to randomly chosen numbers. Furthermore, the choices of K_ℓs are the same for all elements in the same layer, and so this approach is not as distributed and local as the framework of switching networks. The central result underlying the algorithms in [16, 27, 30] was that such random swap-or-not process almost randomly permute any set of $(1-\varepsilon)n$ input elements [16, Theorem 2].

Our results (Theorems 3.5 and 3.6) show that one can replace the swap-or-not scheme defined above by a switching network of logarithmic depth and retain the security properties, as described in [16, 27, 30]. The gain is that one does not need to use a random K_ℓ, which makes the system more robust, and also ensures that the scheme is fully distributed and only local computations are required (since each element needs to follow only its own trajectory, and so, needs to use only $\mathcal{O}(\log n)$ random bits). We note though that we pay some price in our construction: the network used in Theorem 3.6 is more complicated than the original construction of swap-or-not scheme.

The use of additional sources of randomness in switching networks (similar to the use of random K_ℓ in the swap-or-not scheme [16]) has been explored earlier in the literature. For example, it has been shown that if all switches are selected at random, then such a switching network of depth $\mathcal{O}(\log n)$ will almost randomly permute all elements [8, Theorem 1]. This result is incomparable with ours: the randomness coming from the choices of random switches is essential in proving the result in [8]. The key difference between our setting and the setting in [7, 8] is that we prove that almost every *fixed* switching network has some mixing properties; once the network is fixed, the randomness is coming only from random outcomes in the switches.

For further discussions about applications of k-wise almost independent permutations in the context of cryptography and beyond, we refer to [19] and the references therein.

Because of space limitations, we deferred some proofs (in particular, details of our analysis of the coupling in Section 3.1) to the full version of the paper.

2. PRELIMINARIES

We consider the problem of generating an almost uniformly random element from $\mathbb{S}_{n,k}$, or equivalently, an almost random k-partial n-permutation (or an almost $(n-k)$-wise independent permutation). We prove that for any switching network of logarithmic depth with some desired properties, the random shuffling process will almost uniformly generate a random k-partial n-permutation, assuming $k = \Omega(n)$.

2.1 Markov chains and coupling

To analyze the random shuffling process for partial permutations on a switching network $\mathfrak{N}$, we model the process by a Markov chain over the state $\mathbb{S}_{n,k}$, as described in Introduction. Let $\mathfrak{N}$ be a layered network corresponding to a sequence of $\mathfrak{d}$ matchings $\mathbf{M}^{\mathfrak{d}} = (M_0, \ldots, M_{\mathfrak{d}-1}) \in \mathfrak{M}^{\mathfrak{d}}$. If $\pi_0 \in \mathbb{S}_{n,k}$ is the permutation on the input to the network $\mathfrak{N}$, then the Markov chain $(\pi_t)_{t=0}^{\mathfrak{d}}$ of length $\mathfrak{d}$ is defined such that $\pi_{t+1} \in \mathbb{S}_{n,k}$ is obtained from π_t by applying for each matching edge $\{i,j\} \in M_t$ exchanges of the items in nodes i and j independently at random with probability $\frac{1}{2}$.

One can see that in the limit $\mathfrak{d} \to \infty$, the stationary distribution of such Markov chain is almost uniform for almost all networks. Therefore our goal will be to estimate the convergence rate of this Markov chain to the uniform distribution for network $\mathfrak{N}$. To analyze the convergence rate we use the *coupling* approach. While typically Markovian couplings are used to analyze mixing times of Markov chains, in our analysis we heavily rely on *non-Markovian* features of coupling (following the approach initiated in [9]).

Let[1] $\mathfrak{MC} = (\mathcal{Q}_t)_{t\in\mathbb{N}}$ be a discrete-time, possibly time-dependent, Markov chain with a finite state space Ω and a unique stationary distribution $\mu_{\mathfrak{MC}}$. For any random variable X, let $\mathcal{L}(X)$ denote the probability distribution of X, and let $\mathcal{L}(\mathcal{Q}_t \mid \mathcal{Q}_0 = \omega)$ denote the probability distribution of $\mathcal{Q}_t$ given that $\mathcal{Q}_0 = \omega$. We are interested in studying Markov chains for which the statistical distance between $\mathcal{L}(\mathcal{Q}_t \mid \mathcal{Q}_0 = \omega)$ and $\mu_{\mathfrak{MC}}$ tends quickly to zero, independently of $\omega \in \Omega$. To quantify this, we will use the standard measure of the distance between two distributions: the *total variation distance* between two probability distributions $\mathcal{X}$ and $\mathcal{Y}$ over the same finite domain Ω is defined as

$$\begin{aligned} d_{TV}(\mathcal{X},\mathcal{Y}) &= \max_{S\subseteq\Omega} |\mathbf{Pr}_{\mathcal{X}}[S] - \mathbf{Pr}_{\mathcal{Y}}[S]| \\ &= \frac{1}{2}\sum_{\omega\in\Omega} |\mathbf{Pr}_{\mathcal{X}}[\omega] - \mathbf{Pr}_{\mathcal{Y}}[\omega]| \ . \end{aligned}$$

To study the behavior of a Markov chain $\mathfrak{MC}$ with stationary distribution $\mu_{\mathfrak{MC}}$, we define the total variation distance after t steps of $\mathfrak{MC}$ with respect to the initial state $\omega \in \Omega$ as $\Delta_\omega^{\mathfrak{MC}}(t) = d_{TV}(\mathcal{L}(\mathcal{Q}_t \mid \mathcal{Q}_0 = \omega), \mu_{\mathfrak{MC}})$. Then, the standard measure of the convergence of a Markov chain $\mathfrak{MC}$ to its stationary distribution $\mu_{\mathfrak{MC}}$ is the *mixing time*, denoted by $\tau_{\mathfrak{MC}}(\varepsilon)$, which is defined as $\tau_{\mathfrak{MC}}(\varepsilon) = \min\{T \in \mathbb{N} : \forall_{\omega\in\Omega}\forall_{t\geq T}\ \Delta_\omega^{\mathfrak{MC}}(t) \leq \varepsilon\}$.

In this paper we will have $\varepsilon = n^{-c}$ for any constant $c \geq 1$.

Coupling approach.

A *coupling* (see, e.g., [2, 5, 10, 17]) for a Markov chain $\mathfrak{MC} = (\mathcal{Q}_t)_{t\in\mathbb{N}}$ on state space Ω is a stochastic process $(\mathcal{X}_t, \mathcal{Y}_t)_{t\in\mathbb{N}}$ on $\Omega \times \Omega$ such that each of $(\mathcal{X}_t)_{t\in\mathbb{N}}$, $(\mathcal{Y}_t)_{t\in\mathbb{N}}$, considered independently, is a faithful copy of $\mathfrak{MC}$ (i.e., $\mathcal{L}(\mathcal{Q}_t \mid \mathcal{Q}_0 = \omega) = \mathcal{L}(\mathcal{X}_t \mid \mathcal{X}_0 = \omega) = \mathcal{L}(\mathcal{Y}_t \mid \mathcal{Y}_0 = \omega)$ for each $\omega \in \Omega$). The key result on coupling, the so-called *Coupling Inequality* (see, e.g., [2, Lemma 3.6]), states that the total variation distance between $\mathcal{L}(\mathcal{Q}_t \mid \mathcal{Q}_0 = \omega)$ and its stationary distribution $\mu_{\mathfrak{MC}}$ is bounded above by the probability that $\mathcal{X}_t \neq \mathcal{Y}_t$ for the worst choice of initial states $\mathcal{X}_0$

[1] In the text below we will consider infinite Markov chains (which is a standard framework for Markov chains) whereas in our analysis we will only analyze Markov chains on finite length, of length $\mathfrak{d}$. However, since our analysis aims only to show the convergence at the end of the chain after $\mathfrak{d}$ steps, our analysis is equivalent to the analysis of the first $\mathfrak{d}$ steps of a possibly infinite Markov chain.

and $\mathcal{Y}_0$:

$$\max_{\omega\in\Omega}\Delta_{\omega}^{\mathfrak{MC}}(t) \quad \leq \quad \max_{\omega,\omega^*\in\Omega}\mathbf{Pr}[\mathcal{X}_t \neq \mathcal{Y}_t \mid (\mathcal{X}_0,\mathcal{Y}_0)=(\omega,\omega^*)] \ .$$

The classical coupling approach analyzes process $(\mathcal{X}_t,\mathcal{Y}_t)_{t\in\mathbb{N}}$ on the whole space $\Omega\times\Omega$. The *path coupling* method of Bubley and Dyer [5] allows to consider a coupling only for a subset of $\Omega\times\Omega$. Further refinement is coming from an extension of path coupling method called *delayed path coupling* [4, 7, 8]. Comparing to standard coupling, delayed path coupling considers coupling $(\mathcal{X}_t,\mathcal{Y}_t)_{t\in\mathbb{N}}$ with $\mathcal{X}_0$ and $\mathcal{Y}_0$ being similar (like in path coupling [5]), and the goal is to design the coupling by observing the Markov chain in several steps to ensure that for some small t the value of $\mathbf{Pr}[\mathcal{X}_t \neq \mathcal{Y}_t]$ is very small (traditionally path coupling considers only $\mathbf{Pr}[\mathcal{X}_t \neq \mathcal{Y}_t]$ conditioned on $\mathcal{X}_{t-1}$ and $\mathcal{Y}_{t-1}$, whereas delayed path coupling considers $\mathbf{Pr}[\mathcal{X}_t \neq \mathcal{Y}_t]$ conditioned on $\mathcal{X}_0$ and $\mathcal{Y}_0$ only, and thus considers the coupling for multiple steps). We will analyze the convergence of Markov chains using the following lemma.

LEMMA 2.1 (**Delayed Path Coupling Lemma [4, 7, 8]**). *Let $\mathfrak{MC}=(\mathcal{X}_t)_{t\in\mathbb{N}}$ be a discrete-time Markov chain with a finite state space Ω. Let Γ be any subset of $\Omega\times\Omega$. Suppose that there is an integer D such that for every $(\mathcal{X},\mathcal{Y})\in\Omega\times\Omega$ there exists a sequence $\mathcal{X}=\Lambda_0,\Lambda_1,\ldots,\Lambda_r=\mathcal{Y}$, where $(\Lambda_i,\Lambda_{i+1})\in\Gamma$ for $0\leq i<r$, and $r\leq D$. If there exists a coupling $(\mathcal{X}_t,\mathcal{Y}_t)_{t\in\mathbb{N}}$ for $\mathfrak{MC}$ such that for some $T\in\mathbb{N}$, for all $(\mathcal{X},\mathcal{Y})\in\Gamma$, it holds that $\mathbf{Pr}[\mathcal{X}_T\neq\mathcal{Y}_T \mid (\mathcal{X}_0,\mathcal{Y}_0)=(\mathcal{X},\mathcal{Y})]\leq\frac{\varepsilon}{D}$, then*

$$\|\mathcal{L}(\mathcal{X}_T|\mathcal{X}_0=\mathcal{X})-\mathcal{L}(\mathcal{Y}_T|\mathcal{Y}_0=\mathcal{Y})\|\leq\varepsilon$$

for every $(\mathcal{X},\mathcal{Y})\in\Omega\times\Omega$. In particular, $\tau_{\mathfrak{MC}}(\varepsilon/2)\leq T$.

REMARK 2. *The fact that the stationary distribution of Markov chains underlying the process on almost all switching networks is almost uniform, is given here (and will be used frequently throughout the paper) mainly for intuitions and is not needed in the analysis. In fact, as one can see in the statement of Lemma 2.1, our results can be always phrased in the term of the distance between the distributions of any two outputs. That is, for any two inputs π_1, π_2 from Ω, if we apply to them the switching network, then the respective outputs π_1^* and π_2^* will satisfy the following:*

$$\|\mathcal{L}(\pi_1^*|\pi_1)-\mathcal{L}(\pi_2^*|\pi_2)\|\leq\varepsilon \ .$$

(The fact that one can use the uniform distribution μ in the intuitions follows from the fact that if we could prove that $\|\mathcal{L}(\pi_1^*|\pi_1)-\mathcal{L}(\mu)\|\leq\varepsilon/2$ and $\|\mathcal{L}(\pi_2^*|\pi_2)-\mathcal{L}(\mu)\|\leq\varepsilon/2$, then we also would have $\|\mathcal{L}(\pi_1^*|\pi_1)-\mathcal{L}(\pi_2^*|\pi_2)\|\leq\|\mathcal{L}(\pi_1^*|\pi_1)-\mathcal{L}(\mu)\|+\|\mathcal{L}(\pi_2^*|\pi_2)-\mathcal{L}(\mu)\|\leq\varepsilon$, which shows that these two claims are almost equivalent.) Note that in fact our results establish that the stationary distribution of the underlying processes is almost uniform.

Using the Delayed Path Coupling Lemma.

Our goal is to prove that independently of the initial k-partial n-permutation, at the end of the random shuffling process on $\mathfrak{N}$ the obtained k-partial n-permutation will have an almost uniform distribution. We consider a Markov chain of length $\mathfrak{d}$ with the state space $\mathbb{S}_{n,k}$ of all k-partial n-permutations. We define Γ to be the set of all pairs of k-partial n-permutations $\pi_1,\pi_2\in\mathbb{S}_{n,k}$ that differ on exactly two elements:

$$\pi_1(i)=\begin{cases}\pi_2(\ell) & \text{if } i=r \ ,\\ \pi_2(r) & \text{if } i=\ell \ ,\\ \pi_2(i) & \text{otherwise.}\end{cases}$$

Note that for any two $\pi^*,\pi^{**}\in\mathbb{S}_{n,k}$, there is a sequence $\pi^*=\pi_0,\pi_1\ldots\pi_r=\pi^{**}$ with $r\leq n$, such that each pair π_i, π_{i+1} differs on exactly two elements (i.e., $(\pi_i,\pi_{i+1})\in\Gamma$), thus we can use $D=n$ in Lemma 2.1.

Next, for any π_1 and π_2 that differ on exactly two elements, we will define a coupling $(\mathcal{X}_t,\mathcal{Y}_t)_{t=0}^{T}$ with $\mathcal{X}_0=\pi_1$ and $\mathcal{Y}_0=\pi_2$, such that each $\mathcal{X}_{t+1}$ and $\mathcal{Y}_{t+1}$ is obtained from $\mathcal{X}_t$ and $\mathcal{Y}_t$, respectively, by applying a single shuffle M_t in $\mathfrak{N}$. Our goal is to ensure that the designed coupling for the random shuffling process will have $\mathbf{Pr}[\mathcal{X}_T\neq\mathcal{Y}_T]\leq n^{-c}$ for any constant $c\geq 1$ and some $T=\mathcal{O}(\log n)$, $T\leq\mathfrak{d}$. By Lemma 2.1, this will ensure that $\tau_{\mathfrak{MC}}(1/n)\leq T$ for the random shuffling process on $\mathfrak{N}$.

2.2 Modeling (random walks in) $\mathfrak{N}$ by (random walks in) expanders

Let $G=(V,E)$ be a d-regular graph on n vertices and let A_G be its adjacency matrix. The spectrum of G is the spectrum of A_G with its n real eigenvalues: $d=\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_n\geq -d$. Let $\lambda(G)=\max\{|\lambda_2|,|\lambda_n|\}$ and we say a d-regular graph G is an *α-expander* if $\lambda(G)\leq\alpha d$. Intuitively, a graph G is a good expander if $d-\lambda(G)$ is lower bounded by a positive constant (for more information about expanders, see, e.g., [15] and the references therein).

Let us consider a switching network $\mathfrak{N}$ of depth $\mathfrak{d}$ that corresponds to $\mathbf{M}^{\mathfrak{d}}=(M_0,\ldots,M_{\mathfrak{d}-1})\in\mathfrak{M}^{\mathfrak{d}}$. Define an *$\langle\ell,r\rangle$-truncate of* $\mathfrak{N}$ to be the multigraph $G=(V,E)$ on vertex set $V=\{1,\ldots,n\}$ with the edge set E consisting of all pairs (i,j) for which there is a path from i to j in the network induced by $M_\ell,M_{\ell+1},\ldots,M_{\ell+r-1}$; if there are s paths from i to j then we have s edges (i,j) in E. (In other words, if it possible from a vertex i at layer ℓ to reach a vertex j at layer $\ell+r$, then $(i,j)\in E$.) Notice that G is 2^r-regular and it has selfloops (in particular, $(i,i)\in E$ for every i).

We begin with the following lemma about truncate of almost all switching networks.

LEMMA 2.2. *For every $r\geq 4$, there is a constant $\mathfrak{a}$, $0<\mathfrak{a}<1$, such that for almost every switching network $\mathfrak{N}$ (all but at most a $\frac{1}{n^2}$ fraction[2]), for every $0\leq\ell\leq\mathfrak{d}-r$, the $\langle\ell,r\rangle$-truncate G of $\mathfrak{N}$ is an $(1-\mathfrak{a})$-expander.*

PROOF. We prove the lemma only for $r=4$; the extension to arbitrary constant r is straightforward.

For every $i\in\{1,\ldots,n\}$, let us observe that the following are four neighbors of i in G:

- vertex j_0 matched to i in M_ℓ,
- vertex j_1 matched to j_0 in $M_{\ell+1}$,
- vertex j_2 matched to j_1 in $M_{\ell+2}$, and

[2]For simplicity of presentation we will assume that term *"almost all switching networks"* refers to all but at most a $\frac{1}{n^2}$ fraction of all (relevant) networks, though it is easy to extend our claims to hold for all but at most $\frac{1}{n^c}$ fraction of all networks for an arbitrary large constant c.

- vertex j_3 matched to j_2 in $M_{\ell+3}$.

Indeed, there is a path from vertex i at layer ℓ to vertex j_r at layer $\ell+\kappa$, $\kappa \in \{0,1,2,3\}$, and then our construction ensures that since for every s at layer l, there is a path from s to itself at any layer $l' \geq l$, we can conclude that there is also a path in $\mathfrak{N}$ from vertex i at layer ℓ to vertex j_κ at any layer ℓ', $\ell' \geq \ell+\kappa$.

Next, let us consider subgraph $G^{\langle 4\rangle}$ of graph G with edge set containing only the edges (i, j_r) described above, $\kappa \in \{0,1,2,3\}$. Note that $G^{\langle 4\rangle}$ is a subgraph G obtained by taking four perfect matchings $M_\ell, M_{\ell+1}, M_{\ell+2}, M_{\ell+3}$ on $\{1,\ldots,n\}$. It is known that almost every graph H obtained by taking four perfect matchings is a good expander with expansion $\lambda(H) \leq 2\sqrt{3}+\varepsilon \leq 3.5$ (see, e.g., Friedman [12, Theorem 1.3]). Hence, almost every graph $G^{\langle 4\rangle}$ is a $\frac{7}{8}$-expander and therefore there is a positive constant $\mathfrak{a}$ such that G, which is a supergraph of $G^{\langle 4\rangle}$, is almost always an $(1-\mathfrak{a})$-expander. □

Because of our transformation presented above, we will analyze the random shuffling process for any switching network $\mathfrak{N}$ for which each $\langle i\cdot r, r\rangle$-truncate is a good expander. Let us call a switching network $\mathfrak{N}$ to be $\mathfrak{good}$ if there is a constant r and another positive constant $\mathfrak{a}$ such that every $\langle i\cdot r, r\rangle$-truncate is a $(1-\mathfrak{a})$-expander, $0 \leq i < \mathfrak{d}/r$. We have the following two facts.

PROPOSITION 2.3. *Almost all (all but a $\frac{1}{n^2}$ fraction) switching networks of logarithmic depth are* $\mathfrak{good}$.

PROOF. Follows directly from Lemma 2.2. □

PROPOSITION 2.4. *One can explicitly construct a* $\mathfrak{good}$ *switching network* $\mathfrak{N}$.

PROOF. Follows from known results about constructions of good expanders (see, e.g., [15]) using the following approach: Take a known construction of any 3-regular "good expander" $G = (\{1,\ldots,n\}, E)$, e.g., one from [15]. 4-color the edges of G (G is 4-edge-colorable since the maximum degree of G is 3), and then add arbitrarily edges to E to make G a union of four perfect matchings, which we will call PM_0, PM_1, PM_2, PM_3. Then, we define $\mathfrak{N}$ to be a switching network of depth $\mathfrak{d}$, with $\mathfrak{d}$ is divisible by 4, that is defined by a sequence of $\mathfrak{d}$ matchings $M_0,\ldots,M_{\mathfrak{d}-1}$ with $M_{4i+j} = PM_j$ for every $i \in \{0,\ldots,\frac{1}{4}\mathfrak{d}-1\}$ and $j \in \{0,1,2,3\}$. The same arguments as those used in the proof of Lemma 2.2 show that there is a positive constant $\mathfrak{a}$ such that $\langle 4i, 4\rangle$-truncate of $\mathfrak{N}$ is a $(1-\mathfrak{a})$-expander for every $i \in \{0,\ldots,\frac{1}{4}\mathfrak{d}-1\}$. □

From now on, we will assume that every switching network $\mathfrak{N}$ we consider is $\mathfrak{good}$.

Random walks in $\mathfrak{N}$ as random walks in expanders.

In our analysis, we will consider random walks in switching networks, which is the process defined by first taking an arbitrary element at a starting position, and then proceeding through the network from the first layer to the last, taking a random switch between each pair of layers. Let us observe that we can model a random walk in a switching network $\mathfrak{N}$ by taking the starting vertex x and then first moving to a random neighbor of x in $\langle 0, r\rangle$-truncate of $\mathfrak{N}$, then taking next random neighbor in $\langle r, r\rangle$-truncate of $\mathfrak{N}$, and so on, to reach the final layer of $\mathfrak{N}$. If we take $r \geq 4$, then every $\langle i\cdot r, r\rangle$-truncate of $\mathfrak{N}$ is a good expander ($(1-\mathfrak{a})$-expander), and so the random walk in $\mathfrak{N}$ can be modeled by a random walk in a sequence of expanders. If G_i is a $\langle i\cdot r, r\rangle$-truncate of $\mathfrak{N}$, then the random walk of length sr in $\mathfrak{N}$ (assuming the depth of $\mathfrak{N}$ is at least sr) can be modeled by a random walk that takes the first step in G_0, then the next step in G_1, and so on, until performing the last step in G_s. Assuming every G_i is an expander, we can use the theory of random walks in expanders to analyze the behavior of such a random walk. (Observe that the approach above when combined with known basic facts about random walks in expanders immediately implies that for every $\mathfrak{good}$ switching network $\mathfrak{N}$ of depth $\mathfrak{d}$, where $\mathfrak{d} \geq c\log n$ for a sufficiently large positive constant c, after applying the random shuffling process on $\mathfrak{N}$, the distribution of the location of any single element is almost uniform. We notice however that in our analysis more properties of random walks will be needed.)

3. GENERATING PARTIAL PERMUTATIONS USING SWITCHING NETWORKS

In this section we will study the random shuffling process for a switching network $\mathfrak{N}$ of depth $\mathfrak{d}$ corresponding to $\mathbf{M}^{\mathfrak{d}} = (M_0,\ldots,M_{\mathfrak{d}-1}) \in \mathfrak{M}^{\mathfrak{d}}$. We will make two assumptions about $\mathfrak{N}$:

(1) switching network $\mathfrak{N}$ is $\mathfrak{good}$ and

(2) $\mathfrak{d} \geq c\log n$ for a sufficiently large constant c.

Our approach uses the Delayed Path Coupling approach, as outlined at the end of Section 2.1.

Let $k = \Omega(n)$. Let π_1 and π_2 be two arbitrary k-partial n-permutations from $\mathbb{S}_{n,k}$ that differ on precisely two elements. Our goal is to define a coupling $(\mathcal{X}_t, \mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ that satisfies the following conditions:

Initial state: $(\mathcal{X}_0, \mathcal{Y}_0) = (\pi_1, \pi_2)$;

Coupling: each $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ and $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ in isolation is a faithful copy of the random shuffling process on $\mathfrak{N}$;

Convergence: for certain $T = \mathcal{O}(\log n)$, $T \leq \mathfrak{d}$, with high probability: $\mathcal{X}_t = \mathcal{Y}_t$ for all $t \geq T$.

By the Delayed Path Coupling Lemma 2.1, these conditions would imply that the mixing time of the random shuffling process on $\mathfrak{N}$ for generating random k-partial n-permutations is $\mathcal{O}(\log n)$.

We will define the coupling by first allowing the process $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ to be run arbitrarily and then we will set the sequence $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ in a non-Markovian way to ensure the second and the third properties above. By non-Markovian we mean that the sequence $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ will be defined only once the entire sequence $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ is known, and each $\mathcal{Y}_t$ with $t \leq T$ depends on the entire $(\mathcal{X}_t)_{t\leq T}$. (Markovian coupling, which has been more commonly used in the past, would mean that $\mathcal{Y}_{t+1}$ depends only on $\mathcal{Y}_t$, $\mathcal{X}_t$, and $\mathcal{X}_{t+1}$.)

Our analysis follows the approach proposed in [9] for the analysis of Thorp shuffle, though a more complex structure of our networks makes details more challenging. We split the process into two phases, the first phase (Section 3.1.1) corresponding to the first $\mathcal{O}(\log n)$ layers of $\mathfrak{N}$ and the second phase (Section 3.1.2) corresponding to the remaining layers of $\mathfrak{N}$, and the final coupling will be done after seeing all random choices for $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ in these two phases (Section

3.1.3). In Sections 3.1.1 and 3.1.2, we will consider $\mathcal{O}(\log n)$ steps of the random shuffling process on $\mathfrak{N}$.

Notation.

When we consider the process as one of moving the input elements from left to right, that is, by applying first shuffle for M_0, then shuffle for M_1, and so on, then we use term *element* to denote the current status of a given input element, and *position* or *location* to denote its current position in the switching network. If we run the random shuffle process, then each time we have two elements that are connected by a switch (to be swapped at random), then we say that these two elements are a *match* at a given moment.

3.1 Coupling for partial permutations on $\mathfrak{N}$

We first consider the random shuffling process starting at the state $\mathcal{X}_0 = \pi_1$ and we will look at the sequence $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$. Afterwards we will analyze the properties of $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ to define the sequence $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$. Let α and β be the two elements which have distinct positions in π_1 and π_2. Let $\varpi = \lfloor \log_2 k \rfloor$; hence $2^{\varpi} \leq k < 2^{\varpi+1}$ (ϖ be the largest power of 2 not greater than k). Let $\mathbb{B}$ be the set of the 0 elements except possibly for the elements α and β (if either α or β is a 0). Observe that $|\mathbb{B}| \geq k - 1 \geq 2^{\varpi} - 1$. Let $\mathbb{B}^* = \mathbb{B} \cup \{\alpha, \beta\}$.

We will follow the elements from $\mathbb{B}^*$ and those from outside $\mathbb{B}^*$, and every time we have a match involving at most one element from $\mathbb{B}^*$ in $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$, we will make at once the same choices (outcomes of the matches) for both $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ and $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$. Furthermore, for a large part of the outcomes of the matches between two elements from $\mathbb{B}^*$ in $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ we will also set it identically in both copies, in $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ and $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$. However, our main focus is on a small number of appropriately selected matches between pairs of elements from $\mathbb{B}^*$ in $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$. Our analysis is in two stages. We present here the main ideas behind our analysis; more details are deferred to the full version of the paper.

3.1.1 First stage — using fundamental trees

In the first stage, we will try to construct two disjoint full binary trees, which we call *fundamental trees*, that are obtained using the following branching process:

- Create two trees whose roots are the two elements α and β (the elements where $\mathcal{X}_0$ and $\mathcal{Y}_0$ differ).
- Suppose that we have already built the two trees for $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_t$ (i.e., for the first t layers of $\mathfrak{N}$). Let $\varrho = 2^{\varpi-2}$. Let v be any element corresponding to a leaf $\hat{v}$ in one of the trees at depth strictly smaller than $\log_2 \varrho$. If v is to be matched in the t^{th} shuffle (as defined by M_t) to an element u from $\mathbb{B}^*$ and u was not used in the construction of any of the trees before, then we *branch at* $\hat{v}$. In that case, $\hat{v}$ has two new children: one corresponding to the element v and one corresponding to the element u. We perform this operation for all such leaves $\hat{v}$ at the same time, to build two trees for $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_{t+1}$.

We continue this process for increasing t to build two trees until all leaves of both trees are at the same level and each tree has exactly $\varrho = 2^{\varpi-2}$ leaves (which is why we branched only leaves at depth smaller than $\log_2 \varrho$); such full binary trees will be called the *fundamental trees*.

Our central result is the following lemma for $\mathfrak{good}$ switching networks $\mathfrak{N}$ that relies on the structure of expanders and the analysis of random walks on expander graphs.

LEMMA 3.1. *There is a constant c such that if we run the random shuffling process for $c \log_2 n$ steps with all switches set at random then the probability that two fundamental trees will be built is at least $1 - n^{-3}$.*

Let us discuss the intuitions behind this phase and state central properties of our construction. We observe that each fundamental tree has only one element from outside $\mathbb{B}$: either α or β. Therefore, since all but one elements from each fundamental tree are in $\mathbb{B}$ (are 0s), the process of setting the outcomes of the matches in each of the fundamental trees will correspond to the random selection of the position for α (or β) in the tree; since all elements in $\mathbb{B}$ are identical (are 0s), we can arbitrarily permute them without affecting the outcome of the process. Next, we observe that the trees do not depend on the random outcomes of the matches during the branching. In other words, if we have two instances of the random shuffling process such that the second instance differs from the first one only in the (outcome of the) switches defining the branching for the fundamental trees in the first instance, then the second instance have the same fundamental trees (we may only have permuted elements in each of the fundamental trees). Finally, conditioned on the final positions of the leaves in the fundamental trees, the choice of the final positions of α and β is uniformly random. That is, if the tree containing α has the leaves at positions $p_1, \ldots, p_\varrho$, then if we randomly decide the outcomes of the matches during the branching, then for every i, the probability that α will end up at position p_i is $\frac{1}{\varrho}$. The same property holds for β.

3.1.2 Second stage — using fundamental matching

In the second stage, we fix the 2ϱ leaves of the two fundamental trees built in the first stage. Our goal is to show that if we run next $\mathcal{O}(\log n)$ steps of the random shuffling process defining $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ then we can find a set $\mathbb{M}$ of ϱ matches which forms a perfect matching between the leaves of the two fundamental trees. That is, $\mathbb{M}$ is a set of ϱ matches in the random shuffling process, such that for every match $(v, u) \in \mathbb{M}$, v is a leaf from the first tree, u is a leaf from the second tree, and there is no other pair in $\mathbb{M}$ which contains either v or u. Once we have such a matching, then the *lexicographically first* perfect matching between the leaves of the two fundamental trees will be called the *fundamental matching* $\mathbb{M}^*$. (That is, if we number the switches in the way how they are generated in $\mathfrak{N}$ by matchings $M_0, \ldots, M_{\mathfrak{d}-1}$, then for every other matching $\mathbb{M}$ between the leaves of the two fundamental trees there is an index ℓ, such that after having the same matches in $\mathbb{M}^*$ and $\mathbb{M}$ in the first $\ell - 1$ switches, the ℓ^{th} switch has a match in $\mathbb{M}^*$ and no match in $\mathbb{M}$.)

With this machinery at hand, we only need to prove our main structural result, which relies on the analysis of random walks in expanders using the tools (strong hitting property) developed in [14, 18] (see also [15]).

LEMMA 3.2. *Let us fix any two sets of ϱ disjoint positions for the leaves of the fundamental trees. There is a constant c such that if we run the random shuffling process for $c \log_2 n$ steps with all switches set at random then the probability that there is a fundamental matching is at least $1 - n^{-3}$.*

3.1.3 Coupling

Now, we are ready to define the coupling. We split $\mathfrak{N}$ into two switching subnetworks of depth $\mathfrak{d}/2$ each, the first network $\mathfrak{N}_1$ corresponding to $(M_0, \ldots, M_{\mathfrak{d}/2-1}) \in \mathfrak{M}^{\mathfrak{d}/2}$ and the second network $\mathfrak{N}_2$ corresponding to $(M_{\mathfrak{d}/2}, \ldots, M_{\mathfrak{d}-1}) \in \mathfrak{M}^{\mathfrak{d}/2}$. We run the first stage of the random shuffling process for $\mathfrak{N}_1$ and construct two fundamental trees T_1 and T_2 with ϱ leaves each. Then, we run the second stage of the random shuffling process for $\mathfrak{N}_2$ and construct the fundamental matching $\mathbb{M}$ between the leaves of T_1 and T_2.

Let us first consider the scenario that one fails to construct T_1, T_2, and $\mathbb{M}$ in the process described above. By Lemmas 3.1–3.2 we know that this case is very unlikely, and we set the coupling for $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ and $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ to be the identity coupling, i.e., all switches will be set in the same way for both $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ and $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$.

Therefore from now on, we will analyze the scenario that the two fundamental trees T_1 and T_2 and that the fundamental matching $\mathbb{M}$ have been built.

To define the coupling, for both $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ and $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$, we will use *identical* outcomes for all the random choices in the switches that are not involved in the (branching) matches inside the trees T_1 and T_2, and are not among the matches in $\mathbb{M}$. Let L_1 be the set of positions that the leaves of T_1 reach at the end of the first stage and L_2 be the set of positions that the leaves of T_2 reach at the end of the first stage. For simplicity of notation, we assume that every match in $\mathbb{M}$ is of the form (p, q), where p is an element that reaches a position in L_1 at the end of the first stage and q is an element that reaches a position in L_2 at the end of the first stage; in this case, we use the notation $\mathbb{M}(p) = q$ and $\mathbb{M}(q) = p$.

We observe that since T_1 and T_2 are complete binary trees, if all the choices inside the trees T_1 and T_2 have been done at random, then α reaches the position at the end of the first stage that is uniform in L_1, β reaches the position at the end of the first stage that is uniform in L_2, and these positions are independent. With this, we define the coupling for $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}/2}$ and $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}/2}$ in the first stage such that if α reaches position p at the end of the first stage and β reaches position q at the end of the first stage for the sequence $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}/2}$, then we set the random outcomes for the sequence $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}/2}$ in the first stage such that α (which in $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ is traversing through the tree T_2) reaches position $\mathbb{M}(p)$ at the end of the first stage, and β (which in $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ is traversing through T_1) reaches position $\mathbb{M}(q)$ at the end of the first stage for the sequence $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}/2}$.

To define the coupling for the second stage, we perform the same choices in the switches defining $(\mathcal{X}_t)_{t=\mathfrak{d}/2}^{\mathfrak{d}}$ and $(\mathcal{Y}_t)_{t=\mathfrak{d}/2}^{\mathfrak{d}}$ except for the choices for the outcomes of the matches in $\mathbb{M}$, where we use *reverse* choices for the matches in $\mathbb{M}$ for $(\mathcal{Y}_t)_{t=\mathfrak{d}/2}^{\mathfrak{d}}$ than those in $(\mathcal{X}_t)_{t=\mathfrak{d}/2}^{\mathfrak{d}}$.

Now we are ready to state our next key lemma.

LEMMA 3.3. *The process defining $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ and $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ is a proper coupling.*

To use the coupling defined above in the Delayed Path Coupling Lemma 2.1, we have to estimate the probability that in our coupling $\mathcal{X}_t \neq \mathcal{Y}_t$, for large enough $t \in \mathbb{N}$, $t = \Theta(\log n)$. We first observe that without revealing the outcome of the matches in T_1, T_2, and $\mathbb{M}$, the final positions of the elements outside L_1 and L_2 are fixed. Furthermore, since in L_1 and L_2, all elements other than α and β are from $\mathbb{B}$ (and hence all are 0s and thus indistinguishable), we only have to consider the final positions of α and β. Consider first the chain $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}/2}$ and suppose that α finished the first stage at position p and β finished the first stage at position q. In $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}/2}$, the coupling ensures that in the first stage α finishes at position $\mathbb{M}(p)$ and β at position $\mathbb{M}(q)$. Then, the only change in the performance of $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ and $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ is at the two matches $(p, \mathbb{M}(p))$ and $(q, \mathbb{M}(q))$. The key property of our coupling is that since the outcomes of the matches in $\mathbb{M}$ for $(\mathcal{X}_t)_{t=0}^{\mathfrak{d}}$ are reverse with respect to the choices of the matches in $\mathbb{M}$ for $(\mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$, we will have $\mathcal{X}_{\mathfrak{d}} = \mathcal{Y}_{\mathfrak{d}}$ at the end of the second stage.

We can now summarize properties of our coupling for the random shuffling process in a $\mathfrak{good}$ network $\mathfrak{N}$:

- If T_1, T_2, and $\mathbb{M}$ have been successfully constructed then our coupling $(\mathcal{X}_t, \mathcal{Y}_t)_{t=0}^{\mathfrak{d}}$ ensures that $\mathcal{X}_T = \mathcal{Y}_T$ for all T with $T_0 \leq T \leq \mathfrak{d}$, where $T_0 = \mathcal{O}(\log n)$, and
- T_1, T_2, and $\mathbb{M}$ have been successfully constructed with probability at least $1 - 2n^{-3}$ (by Lemmas 3.1–3.2).

3.2 Final results

We can combine our analysis above with the Delayed Path Coupling Lemma 2.1 to conclude the following results about random generations of k-partial n-permutations $\mathbb{S}_{n,k}$.[3]

THEOREM 3.4. *Let $k = \Omega(n)$. Let $\mathfrak{N}$ be a $\mathfrak{good}$ switching network of depth $\mathfrak{d}$ with $\mathfrak{d} \geq c \log n$, for a sufficiency large constant c. Then $\mathfrak{N}$ generates random k-partial n-permutations almost uniformly.*

That is, for any positive constant c_1, if $\pi \in \mathbb{S}_{n,k}$ is the permutation generated by the switching network $\mathfrak{N}$ on an arbitrary input from $\mathbb{S}_{n,k}$ and μ is the uniform distribution over $\mathbb{S}_{n,k}$, then $d_{TV}(\mathcal{L}(\pi), \mu) \leq \mathcal{O}(n^{-c_1})$.

The following are direct implications of our results and of Propositions 2.3 and 2.4.

THEOREM 3.5. *For any $\varepsilon > 0$, almost every (all but a $\mathcal{O}(n^{-2})$ fraction) switching network $\mathfrak{N}$ of depth $\mathfrak{d}$ ($\mathfrak{d} \geq c \log n$, for a sufficiency large constant c) almost randomly permutes any set of $(1-\varepsilon)n$ elements.*

That is, for any positive constant c_1, if π is the permutation generated by the switching network $\mathfrak{N}$ on an arbitrary input εn-partial n-permutation, and μ is the uniform distribution over all εn-partial n-permutations, then $d_{TV}(\mathcal{L}(\pi), \mu) \leq \mathcal{O}(n^{-c_1})$.

THEOREM 3.6. *For any $\varepsilon > 0$, there is an explicit switching network $\mathfrak{N}$ of depth $\mathfrak{d}$ ($\mathfrak{d} \geq c \log n$, for a sufficiency large constant c) that almost randomly permutes any set of $(1-\varepsilon)n$ elements.*

That is, for any positive constant c_1, if π is the permutation generated by the switching network $\mathfrak{N}$ on an arbitrary input εn-partial n-permutation, and μ is the uniform distribution over all εn-partial n-permutations, then $d_{TV}(\mathcal{L}(\pi), \mu) \leq \mathcal{O}(n^{-c_1})$.

[3]Let us remind that in the analysis we have been aiming to achieve the error term to be of the form $\mathcal{O}(n^{-2})$, however, it is not difficult to see that the analysis can be extended in a straightforward way to achieve the error term $\mathcal{O}(n^{-c_1})$, for any constant c_1.

It is not difficult to see that this result yields also the result for permuting 0s and 1s (see also [7, 29]).

THEOREM 3.7. *Almost every switching network $\mathfrak{N}$ of depth $\mathfrak{d}$ ($\mathfrak{d} \geq c \log n$, for a sufficiency large constant c) almost randomly permutes any sequence of n 0s and 1s. The same result holds for every* good *switching network $\mathfrak{N}$ of depth $\mathfrak{d} \geq c \log n$.*

That is, for any constant c_1 and for any s, $0 \leq s \leq n$, if π is the permutation generated by the switching network $\mathfrak{N}$ above when applied to the input consisting of s 0s and $n-s$ 1s, and μ is the uniform distribution over all permutations of s 0s and $n-s$ 1s, then $d_{TV}(\mathcal{L}(\pi), \mu) \leq \mathcal{O}(n^{-c_1})$.

PROOF. Let us consider any input sequence of n 0s and 1s, and let s be the number of 0s, and hence $n-s$ be the number of 1s. Without loss of generality, let $s \geq \frac{n}{2}$ (if $s < \frac{n}{2}$ then swap 0s and 1s). Then the first result follows from Theorem 3.4 with $k = s$ and the second results follows from Theorem 3.5 with $\varepsilon n = s$. □

REMARK 3. *As mentioned earlier in our discussion about the stationary distribution of the underlying Markov chain (cf. Remark 2), our results can be also represented in a way independent of the stationary distribution. The results above (Theorem 3.4 – 3.7) can be also read as that for any pair π_1, π_2 from the domain input (the same for both π_1 and π_2), if π_1^* and π_2^* is the output of applying switching network $\mathfrak{N}$ to π_1 and π_2, respectively, then $d_{TV}(\mathcal{L}(\pi_1^*), \mathcal{L}(\pi_2^*)) \leq \mathcal{O}(n^{-c_1})$.*

3.2.1 Arguments behind Theorem 1.1

Let us give arguments behind the proof of Theorems 1.1, which follows from our result in Theorem 3.4. Let us state a more formal version of that theorem first. Let $\mathbb{S}_n$ be the set of all n-permutations, $\mathbb{S}_n = \mathbb{S}_{n,0}$.

THEOREM 1.1' *Let c_2 be an arbitrary constant. There is an explicit switching network $\mathfrak{N}$ of depth $\mathcal{O}(\log^2 n)$ and with $\mathcal{O}(n \log n)$ switches such that if $\pi \in \mathbb{S}_n$ denotes the permutation generated by $\mathfrak{N}$ and μ is the uniform distribution over $\mathbb{S}_n$, then $d_{TV}(\mathcal{L}(\pi), \mu) \leq \mathcal{O}(n^{-c_2})$.*

PROOF. We follow the approach used earlier in a similar context by Morris and Rogaway [27].

We will define switching network $\mathfrak{N}$ by $\mathcal{O}(\log n)$ switching networks $\mathfrak{N}_1, \ldots, \mathfrak{N}_{\lfloor \log_2 n \rfloor}$, applied one after the other. Network $\mathfrak{N}_1$ will have n inputs and n outputs. Network $\mathfrak{N}_2$ will take as its inputs the first $\lfloor n/2 \rfloor$ outputs from $\mathfrak{N}_1$, and will permute them to obtain $\lfloor n/2 \rfloor$ outputs, leaving the outputs of $\mathfrak{N}_1$ with locations $\lfloor n/2 \rfloor + 1 \ldots n$ untouched. In general, for any ℓ, $2 \leq \ell \leq \lfloor \log_2 n \rfloor$, network $\mathfrak{N}_\ell$ will take as its inputs the first $\mathfrak{s}_\ell$ outputs from $\mathfrak{N}_{\ell-1}$, and will permute them to obtain $\mathfrak{s}_\ell$ outputs, leaving the outputs of $\mathfrak{N}_{\ell-1}$ with locations $\mathfrak{s}_\ell + 1 \ldots n$ untouched, where $\mathfrak{s}_\ell$ is defined recursively as $\mathfrak{s}_1 = n$ and $\mathfrak{s}_\ell = \lfloor \mathfrak{s}_{\ell-1}/2 \rfloor$ for $\ell \geq 2$. (Note that $\mathfrak{s}_\ell = \Theta(n 2^{-\ell})$.)

Before we will proceed, let us observe the following feature of our coupling analysis in Section 3.1. Our analysis shows that for a switching network with N inputs we can design a coupling that will succeed with probability at least $1 - \mathcal{O}(N^{-2})$, which, as we have argued before, could be made $1 - p_N$, with $p_N = \mathcal{O}(N^{-c_1})$ for an arbitrary constant c_1. Now, observe that if we applied the network twice, then the probability of the coupling would be at least $1 - p_N^2$. And in general, if we defined a switching network to consist of r original networks, put one after the other, then the probability of the coupling would be $1 - p_N^r$.

Using the observation above, we will now define networks $\mathfrak{N}_\ell$. $\mathfrak{N}_1$ will be defined as the explicit good switching network with n inputs, as constructed in Theorem 3.5. Each switching network $\mathfrak{N}_\ell$, $2 \leq \ell \leq \lfloor \log_2 n \rfloor$, will be obtained by applying one after the other $\mathfrak{r}_\ell = \Theta(\log n / \log \mathfrak{s}_\ell) = \Theta(\frac{\log n}{\log n - \ell})$ times an explicit good switching network $\mathfrak{N}^{\langle \ell \rangle}$ with $\mathfrak{s}_\ell$ inputs (noting that the remaining inputs in locations $\mathfrak{s}_\ell + 1 \ldots n$ will be connected directly to the outputs with the respective locations $\mathfrak{s}_\ell + 1 \ldots n$). (The number of repetition $\mathfrak{r}_\ell$ is set to ensure that $p_{\mathfrak{s}_\ell}^{\mathfrak{r}_\ell} = \mathcal{O}(n^{-c_1})$, where $p_{\mathfrak{s}_\ell}$ is the probability of the failure to obtain the coupling in Theorem 3.5, when applied to a good switching network with $\mathfrak{s}_\ell$ inputs.)

Therefore, the total depth of $\mathfrak{N}$ is equal to $\sum_{\ell=1}^{\lfloor \log_2 n \rfloor} \mathfrak{r}_\ell \cdot \mathcal{O}(\log \mathfrak{s}_\ell) = \sum_{\ell=1}^{\lfloor \log_2 n \rfloor} \mathcal{O}(\log n) = \mathcal{O}(\log^2 n)$, and the total number of switches in $\mathfrak{N}$ is equal to $\sum_{\ell=1}^{\lfloor \log_2 n \rfloor} \mathfrak{r}_\ell \cdot \mathcal{O}(\mathfrak{s}_\ell \log \mathfrak{s}_\ell) = \sum_{\ell=1}^{\lfloor \log_2 n \rfloor} \mathcal{O}(\mathfrak{s}_\ell \log n) = \sum_{\ell=1}^{\lfloor \log_2 n \rfloor} \mathcal{O}(n 2^{-\ell} \log n) = \mathcal{O}(n \log n)$.

For any permutation $\pi \in \mathbb{S}_n$, let $(\pi)_i$ be the i^{th} element in the permutation π, and for any $i \leq j$, let $\pi^{\langle i,j \rangle} = ((\pi)_i, (\pi)_{i+1}, \ldots, (\pi)_j)$. For any two permutations $\pi, \pi^* \in \mathbb{S}_n$, we say that π and π^* are *consistent on the interval* $[i, j]$ if $\pi^{\langle i,j \rangle} = \pi^{* \langle i,j \rangle}$. Let π_0 be an arbitrary input permutation and let π_ℓ be the permutation obtained after applying networks $\mathfrak{N}_1, \ldots, \mathfrak{N}_\ell$ to π_0. Our construction ensures that $\pi_{\ell+1}$ and π_ℓ are consistent on $[\mathfrak{s}_\ell + 1, n]$, i.e., $\pi_{\ell+1}^{\langle \mathfrak{s}_\ell + 1, n \rangle} = \pi_\ell^{\langle \mathfrak{s}_\ell + 1, n \rangle}$.

Consider the distribution of the output of any switching network $\mathfrak{N}_\ell$, $1 \leq \ell \leq \lfloor \log_2 n \rfloor$, with $\mathfrak{s}_\ell$ inputs (and ignore the inputs $\mathfrak{s}_\ell + 1 \ldots n$, since they are remaining unchanged). We claim that the set of the first $\mathfrak{s}_{\ell+1}$ elements in the output is an almost random subset (of size $\mathfrak{s}_{\ell+1}$) of the $\mathfrak{s}_\ell$ input elements, and the remaining $\mathfrak{s}_\ell - \mathfrak{s}_{\ell+1}$ output elements are almost randomly permuted. To see this, let us consider the reverse process (traversing $\mathfrak{N}_\ell$ from the right to the left), and mark the first $\mathfrak{s}_{\ell+1}$ elements in π_ℓ as 0s, and then the element $\mathfrak{s}_{\ell+1} + i$ as i, $1 \leq i \leq \mathfrak{s}_\ell - \mathfrak{s}_{\ell+1}$. By Theorem 3.4, the starting permutation will be an almost random $\mathfrak{s}_{\ell+1}$-partial $\mathfrak{s}_\ell$-permutation; i.e., the distribution of the 0s is an almost random distribution among the inputs, and the distribution of non-zeros is almost random among the inputs.

Therefore, if we map this claim to the distribution of π_ℓ, then we obtain that for any $\pi', \pi'' \in \mathbb{S}_n$ that are consistent on $[\mathfrak{s}_\ell + 1, n]$, we have $d_{TV}(\mathcal{L}(\pi_\ell^{\langle \mathfrak{s}_{\ell+1}+1, \mathfrak{s}_\ell \rangle} | \pi_{\ell-1} = \pi'), \mathcal{L}(\pi_\ell^{\langle \mathfrak{s}_{\ell+1}+1, \mathfrak{s}_\ell \rangle} | \pi_{\ell-1} = \pi'')) \leq \mathcal{O}(n^{-c_1})$. With this, it is not difficult to see (see [27, Corollary 2] for more detailed arguments) that since at the end of $\mathfrak{N}_1$, the distribution of the last $\mathfrak{s}_2 - \mathfrak{s}_1$ elements differs from the uniform by at most $\mathcal{O}(n^{-c_1})$, since at the end of $\mathfrak{N}_2$, the distribution (conditioned on $\pi_1^{\langle \mathfrak{s}_2+1, \mathfrak{s}_1 \rangle}$) of the next $\mathfrak{s}_3 - \mathfrak{s}_2$ elements differs from the uniform by at most $\mathcal{O}(n^{-c_1})$, and so on, we obtain that for any permutation $\pi \in \mathbb{S}_n$,

$$d_{TV}(\mathcal{L}(\pi_{\lfloor \log_2 n \rfloor} | \pi_0 = \pi), \mu) \quad \leq \quad \mathcal{O}(\log n \cdot n^{-c_1}) \ .$$

Therefore, we set $c_2 = c_1 - 1$ to conclude the theorem. □

4. FINAL COMMENTS

In this paper we show that almost every switching network of logarithmic depth can be used to almost randomly permute any set of $(1 - \varepsilon)n$ elements with any $\varepsilon > 0$. Furthermore, we show that the result still holds for every switch-

ing network of logarithmic depth that has some special expansion properties, leading to an explicit construction of such networks. Our results are obtained using a non-trivial non-Markovian coupling approach to study mixing times of Markov chains which allows us to reduce the problem to some random walk-like problem on expanders.

The central open problem left in this paper is whether one can extend our results to $\varepsilon = 0$, that is, whether one can show that almost every switching network of logarithmic depth can be used to almost randomly permute any set of n elements, that is, to generate an almost random permutation. We conjecture that this claim is true. We would be also interested in explicitly constructing a switching network of logarithmic depth that can generate an almost random permutation. The techniques used in this paper seem to be too weak to attack these problems and we do not know of any straightforward reduction from randomly permuting k-partial n-permutations for $k \geq 0.01n$ to randomly permuting permutations.

5. ACKNOWLEDGMENTS

This paper is dedicated to the memory of my friend and collaborator Berthold Vöcking, who sadly passed away in 2014.

6. REFERENCES

[1] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $cn \log n$ parallel steps. *Combinatorica*. 3: 1–19, 1983.

[2] D. Aldous. Random walks of finite groups and rapidly mixing Markov chains. In *Séminaire de Probabilités XVII 1981/82*, Springer Verlag, Lecture Notes in Mathematics 986, Berlin, 1983, 243–297.

[3] D. Aldous and P. Diaconis. Shuffling cards and stopping times. *American Mathematical Monthly*, 93:333–347, 1986.

[4] P. Berenbrink, A. Czumaj, A. Steger, and B. Vöcking. Balanced allocations: The heavily loaded case. *SIAM Journal on Computing*, 35(6): 1350–1385, March 2006.

[5] R. Bubley and M. Dyer. Path coupling: A technique for proving rapid mixing in Markov chains. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, pp. 223–231, 1997.

[6] R. Bubley and M. Dyer. Faster random generation of linear extensions. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 350–354, 1998.

[7] A. Czumaj, P. Kanarek, M. Kutyłowski, and K. Loryś. Delayed path coupling and generating random permutations via distributed stochastic processes. In *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 271–280, 1999.

[8] A. Czumaj and M. Kutyłowski. Delayed path coupling and generating random permutations. *Random Structures and Algorithms* 17(3–4): 238–259, 2000.

[9] A. Czumaj and B. Vöcking. Thorp shuffling, butterflies, and non-Markovian couplings. In *Proc. 41st Annual International Colloquium on Automata, Languages and Programming*, pp. 344–355, 2014.

[10] P. Diaconis. *Group Representations in Probability and Statistics.* Lecture Notes Monograph Series, Vol. 11, Institute of Mathematical Statistics, 1988.

[11] M. Dyer and C. Greenhill. A genuinely polynomial-time algorithm for sampling two-rowed contigency tables. In *Proc. 25th Annual International Colloquium on Automata, Languages and Programming*, pp. 339–350, 1998.

[12] J. Friedman. A proof of Alon's second eigenvalue conjecture and related problems. *Memoirs of the American Mathematical Society*, 195(910), 2008.

[13] E. Gelman and A. Ta-Shma. The Benes network is $\frac{q(q1)}{2n}$ almost q-set-wise independent. In *Proc. 34th Foundations of Software Technology and Theoretical Computer Science*, pp. 327–338, 2014.

[14] A. Healy. Randomness-efficient sampling within NC^1. *Computational Complexity* 17(1): 3–37, 2008.

[15] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

[16] V. T. Hoang, B. Morris, and P. Rogaway. An enciphering scheme based on a card shuffle. In *Proc. CRYPTO 2012*, pp. 1–13, 2012.

[17] M. Jerrum. Mathematical foundations of the Markov chain Monte Carlo method. In *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 116–165. Springer, 1998.

[18] N. Kahale. Eigenvalues and expansion of regular graphs. *Journal of the ACM* 42(5): 1091–1106, 1995.

[19] E. Kaplan, M. Naor, and O. Reingold. Derandomized constructions of k-wise (almost) independent permutations. Algorithmica 55: 113-133, 2009.

[20] D.E. Knuth. *The Art of Computer Programming.* Volume 3: Sorting and Searching. Third Edition. Addison-Wesley, 1997.

[21] T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers, 1991.

[22] T. Leighton and C. G. Plaxton. Hypercubic sorting networks. *SIAM Journal on Computing*, 27:1–47, 1998.

[23] B. Morris. The mixing time of the Thorp shuffle. *SIAM Journal on Computing*, 38(2): 484–504, 2008.

[24] B. Morris. Improved mixing time bounds for the Thorp shuffle and L-reversal chain. *Annals of Probabability*, 37(2): 453–477, 2009.

[25] B. Morris. Improved mixing time bounds for the Thorp shuffle. *Combinatorics, Probability and Computing*, 22, 118–132, 2013.

[26] B. Morris, P. Rogaway, and T. Stegers. How to encipher messages on a small domain. In *Proc. CRYPTO 2009*, pp. 286–302, 2009.

[27] B. Morris and P. Rogaway. Sometimes-recurse shuffle — almost-random permutations in logarithmic expected time. In *Proc. EUROCRYPT 2014*, pp. 311–326, 2014.

[28] M. Naor and O. Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *Journal of Cryptology*, 12(1): 29–66, 1999.

[29] C. Rackoff and D. R. Simon. Cryptographic defense against traffic analysis. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, 1993.

[30] T. Ristenpart and S. Yilek. The Mix-and-Cut shuffle: Small-domain encryption eecure against N queries. In *Proc. CRYPTO 2013*, pp 392–409, 2013.

[31] D. R. Simon. Private communication, October 1997.