Delayed Path Coupling and Generating Random Permutations via Distributed Stochastic Processes *

Artur Czumaj† Przemka Kanarek§ Mirosław Kutyłowski§‡ Krzysztof Loryś§

Abstract

We analyze various stochastic processes for generating permutations almost uniformly at random in distributed and parallel systems. All our protocols are simple, elegant and are based on performing disjoint transpositions executed in parallel. The challenging problem of our concern is to prove that the output configurations in our processes reach almost uniform probability distribution *very rapidly*, i.e. in a (low) polylogarithmic time.

For the analysis of the aforementioned protocols we develop a novel technique, called *delayed path coupling*, for proving rapid mixing of Markov chains. Our approach is an extension of the *path coupling* method of Bubley and Dyer.

We apply delayed path coupling to three stochastic processes for generating random permutations. For one process, we apply standard (though non-trivial) coupling arguments, for the second one we *construct* a *non-Markovian* coupling, and for the third one we *prove the existence* of a *non-Markovian* coupling. To the best of our knowledge, these are the first non-trivial applications of non-Markovian coupling for proving rapid mixing of Markov chains.

We apply our analysis in diverse areas. We develop a simple permutation network of a polylogarithmic depth generating permutations with almost uniform distribution. A simple EREW PRAM algorithm generating random permutations in time $\mathcal{O}(\log\log n)$ with $\mathcal{O}(n\log^{\mathcal{O}(1)} n)$ processors follows. We improve technique of cryptographic defense against traffic analysis by showing that the underlying stochastic process converges in time $\mathcal{O}(\log n)$ (instead of polylogarithmic time) and thereby dramatically reduce the size of a security parameter. Other applications include load sharing in distributed systems, design of fault resistant systems, and parallel algorithms generating random permutations for models with restricted concurrent writes/reads.

1 Introduction.

Mixing a collection of items is a very frequently used generic process in various applications. This may be, for example, relocation of tasks in a distributed system, or a redistribution of the input to avoid malicious configurations. Mixing has also applications in computer security, when we aim to hide the original structure of the input or the flow of information transmitted. The most fundamental problem related to mixing is that of *random permuting*: there are n items located in n positions; we have to relocate them at random so that each position is given exactly one item and the resulting allocation has (almost) uniform probability distribution over $\mathbb{S}_n$, the set of permutations of n items.

Many algorithms for generating random permutations in parallel environment/systems have been proposed. However, they typically require complicated and powerful tools, like, for example, strong resolution rules for a concurrent access to the shared memory [19, 23] or powerful random number generators [9]. Really practical solutions have been missing.

One typical and intensively studied sequential method of permuting at random is to repeat independent steps, during which we pick one item and move it to some other (e.g. random) position. Such processes correspond to Markov chains. Crucial for their performance is the question how many steps are necessary until probability distribution for configuration of items becomes close to the stationary distribution of the chain, which is uniform. This research, located on a frontier between computer science and theory of stochastic processes, resulted in many deep results (see, e.g., [1, 2, 10]).

In order to design fast and practical algorithms for random permuting in distributed and parallel environment/systems, we may follow a similar strategy and relocate many items in parallel. However, we admit only operations which are easy to implement in such systems. Therefore, we use only *disjoint transpositions*. In the paper, we present three algorithms (and analyze three different stochastic processes) that follow this approach. The first one, called *matching exchange*, is a generalization of the sequential process described. The second one is designed on a fixed switching network. The last one yields good results for almost all switching networks of a logarithmic depth.

Although the algorithms presented in this paper are quite intuitive and easy to formulate, it is challenging to show quality of their output. We describe these processes as Markov chains and concentrate on

*Extended abstract, partially supported by KBN grants 8 S503 002 07, 8 T11C 032 15, ALCOM EU ESPRIT Long Term Research Project 20244 (ALCOM-IT), DFG-Sonderforschungsbereich 376 "Massive Parallelität", and DFG Leibniz Grant Me872/6-1.

†Heinz Nixdorf Institute and Department of Mathematics & Computer Science, University of Paderborn.

‡Work partially done while the author was with the University of Paderborn.

§Institute of Computer Science, University of Wrocław.

proving rapid mixing property of the corresponding stochastic processes. There are quite a few methods to analyze convergence rates of Markov chains: coupling, stopping time, finite Fourier method, the canonical path approach, and the conductance argument (cf. [1, 2, 10, 20, 21, 27] for survey expositions). Even if the canonical path approach and the conductance technique were used very successfully to provide polynomial-time bounds for the mixing times of various, often very complicated Markov chains [11, 20, 21, 27], the obtained bounds seem usually to be far from optimal. Moreover, this research has dealt almost exclusively with "sequential" processes and mixing times polynomial in n. We are aware of very few results on "parallel" processes with polylogarithmic mixing times (see [10, 24, 28]). Our main technical contribution is development of a so called *delayed path coupling* technique for proving (very) rapid mixing of Markov chains, which we use to analyze accurately our algorithms. Delayed path coupling is a refinement of the classical coupling arguments (see, e.g., [1]) and the path coupling technique [5]. The main idea is to use non-Markovian coupling and delay the decisions about current coupling waiting for some future events. This provides much more decision freedom than in the classical case.

Even if it was known that coupling (and path coupling, too) can be used successfully to analyze highly symmetric Markov chains, the processes that we study in the paper did not seem to posses efficient coupling proofs. For example, matching exchange is a "parallel" version of the classical "sequential" Markov chain generating random permutations via random transpositions (cf. [1, 2, 10]). Although the last process is known to converge after $(\frac{1}{2} + o(1))n \ln n$ steps [10], the best known convergence rate obtained by coupling arguments is $\Theta(n^2)$ [1]. While this suggests not to use coupling for our purposes, we show that delayed path coupling provides an accurate analysis of matching exchange process. (We remark that our arguments deliver the first coupling-based proof of the convergence rate of $\mathcal{O}(n \ln n)$ for the classical "sequential" Markov chain mentioned.)

Organization of the paper. In Section 2, we present the main technical results concerning mixing time of stochastic processes that we use. In Section 3, we apply these results in diverse areas. Section 4 provides technical tools used in our analysis and describes the delayed path coupling method. In Section 5, we sketch the proof of Theorem 2.1. Full technical details of this proof are included in Section 6. Proofs of the other main results, Theorems 2.2 and 2.3 are deferred to the full version of the paper.

2 Stochastic processes.

2.1 Non-oblivious exchanges. The first stochastic protocol, called **distributed mixing**, consists of several identical steps performed repeatedly. A step consists of four substeps executed independently by n persons, each holding a single item:

1. Each person tosses a coin and accordingly decides to be either *active* or *passive* during this step.
2. Each active person chooses i.u.r. (*independently and uniformly at random*) a partner.
3. If A has chosen P, then P and *A accepts* themselves if and only if P is passive and A is the only person who has chosen P.
4. Each pair of partners that accepted themselves tosses independently a coin and accordingly either *exchange* the items or keep them.

Distributed mixing has many features important for practical applications: the protocol is extremely simple; no kind of global control is used; it is symmetric: each person behaves in the same way; it is time-homogeneous: each step is executed in the same way and independently of the history of the protocol.

Distributed mixing is a special case of a more general Markov chain. Namely, let G be a complete undirected graph with n vertices numbered $\{1, \ldots, n\}$ each storing a single item. We consider a Markov chain $\mathfrak{M} = (\Pi_t)_{t \in \mathbb{N}}$, called **matching exchange**, on the set $\mathbb{S}_n$ of all permutations of $\{1, \ldots, n\}$ defined by the following transition rule $\Pi_t \mapsto \Pi_{t+1}$: First we choose $k_{t+1} \leq \frac{n}{2}$ according to some probability distribution κ on $\{0, \ldots, \frac{n}{2}\}$ with $\mathbf{E}[\kappa] = \Theta(n)$. Then we pick i.u.r. a matching $\mathbf{M}_{t+1}$ in G of size k_{t+1}, and independently for each edge of $\mathbf{M}_{t+1}$, toss a coin. The edges, where we get tails are called *active edges* and form a matching $\tilde{\mathbf{M}}_{t+1} \subseteq \mathbf{M}_{t+1}$. Then we exchange the items lying on the endpoints of every edge of $\tilde{\mathbf{M}}_{t+1}$, (i.e., if $\{i, j\} \in \tilde{\mathbf{M}}_{t+1}$, then $\Pi_{t+1}(\Pi_t^{-1}(i)) = j$ and $\Pi_{t+1}(\Pi_t^{-1}(j)) = i$).

One can easily verify that $\mathfrak{M}$ is ergodic and symmetric, and therefore that it has the unique stationary distribution which is uniform over $\mathbb{S}_n$ (cf. [20, 21, 27]).

To express the rate of convergence of stochastic processes, we use a standard measure of discrepancy between two probability distributions ϑ and ν on a space Ω, the *variation distance*, defined as

$$\|\vartheta - \nu\| = \tfrac{1}{2} \sum_{\omega \in \Omega} |\vartheta(\omega) - \nu(\omega)| \ .$$

It is easy to see that $\Omega(\log n)$ steps of matching exchange are required to move all the items from their original positions with probability at least $\frac{1}{2}$. We show that, actually, $O(\log n)$ steps are enough to reach a random allocation of all items:

THEOREM 2.1. *There is $T = \mathcal{O}(\log n)$ such that the variation distance between the probability distribution of Π_T and the uniform distribution over $\mathbb{S}_n$ is $\mathcal{O}(\frac{1}{n})$.*

2.2 Oblivious exchanges – switching networks. For the matching exchange protocol, one of sources of randomness is the choice of the matchings used. Can we generate random permutations in the matching-exchange protocol, if the sequence of matchings is fixed? Note that in this case we are talking about a *switching network*. So far, no *explicit* construction of such a switching network with small depth has been known.

THEOREM 2.2. *One can construct a switching network of depth $\log^{\mathcal{O}(1)} n$ with a simple architecture (a recursive construction where a basic block is composed from pipelined butterflies with some levels deleted) that generates permutations of n elements for which the variation distance between the probability distribution obtained from the uniform distribution on $\mathbb{S}_n$ is $\mathcal{O}(\frac{1}{n})$.*

Rackoff and Simon [24] consider the problem of generating random permutations for a fixed but random collection of matchings: Let $\mathbb{M}^\tau$ be the set of all sequences $(M_0, \ldots, M_{\tau-1})$ such that each M_t is a perfect matching of $\{1, \ldots, n\}$. For a given $\mathbf{M}^\tau = (M_0, \ldots, M_{\tau-1}) \in \mathbb{M}^\tau$, consider Markov chain $(\mathbb{B}_t)_{t=0}^{\tau}$ on state space of all 0–1 sequences with $\frac{n}{2}$ zeros and $\frac{n}{2}$ ones with the transition rules $\mathbb{B}_t \mapsto \mathbb{B}_{t+1}$ that for each matching edge $\{i, j\} \in M_t$ exchanges the items in vertices i and j independently with probability $\frac{1}{2}$.

Rackoff and Simon [24, Theorem 3.1] show that after a *polylogarithmic number* of steps of process $\mathbb{B}$ any sequence of 0s and 1s will be almost randomly shuffled for almost every choice of matchings $\mathbb{M}$. They apply this result to show security of a cryptographic protocol (see Section 3.3).

Delayed path coupling provides a simpler proof of convergence of process $\mathbb{B}$. Moreover, we reduce the bound on the mixing time from polylogarithmic (with a two-digit degree [26]) to $\mathcal{O}(\log n)$:

THEOREM 2.3. *There is a $\tau = \mathcal{O}(\log n)$ such that for almost every $\mathbf{M}^\tau \in \mathbb{M}^\tau$ (i.e., for all but $n^{-\Omega(1)} \cdot |\mathbb{M}^\tau|$ many elements from $\mathbb{M}^\tau$) and independently of $\mathbb{B}_0$, the variation distance between the probability distribution of $\mathbb{B}_\tau$ and the uniform distribution on all 0–1 sequences with $\frac{n}{2}$ zeros and $\frac{n}{2}$ ones is $\mathcal{O}(\frac{1}{n})$.*

In particular, Theorem 2.3 implies *existence* of switching networks of depth $\mathcal{O}(\log^2 n)$ that generate permutations of n elements so that the variation distance between the probability distribution obtained and the uniform distribution on $\mathbb{S}_n$ is $\mathcal{O}(\frac{1}{n})$.

3 Applications.

3.1 Parallel algorithms generating random permutations. The stochastic processes considered in this paper lead directly to a number of novel algorithms for generating random permutations on parallel machines. Our first application is the following, rather unexpected result.

THEOREM 3.1. *Random permutations of n items can be generated in $\mathcal{O}(\log\log n)$ time on the EREW PRAM with $n \log^{\mathcal{O}(1)}(n)$ processors so that the variation distance between the probability distribution of the permutations generated and the uniform probability distribution over $\mathbb{S}_n$ is $\mathcal{O}(\frac{1}{n})$.*

Theorem 3.1 follows immediately from Theorem 2.2 (the non-uniform version follows also from Theorem 2.3 as well as from [24, Theorem 3.1]): in one step, EREW PRAM generates a random setting of the switches of the network given by Theorem 2.2. Then it finds the paths leading from the input nodes to the output nodes using pointer doubling.

Since there are extremely few algorithms on the EREW PRAM with sublogarithmic runtimes, it is quite surprising that double logarithmic time has been achieved for such a complicated problem as generating random permutations. The previous EREW PRAM algorithms for this problem have runtime $\Omega(\log n)$ [9, 19]. Our algorithm beats even the best previously known CREW PRAM algorithm [9] regarding the total work (while preserving the runtime).

The total work of the EREW PRAM algorithm is $\mathcal{O}(n \log^{\mathcal{O}(1)} n)$. We can reduce it on other parallel computation models and by using distributed mixing instead of the network from Theorem 2.2. For instance, a direct implementation of distributed mixing seems to be the method of choice for the OCPC machine (*Optical Communication Parallel Computer*) (see [4, 17] for a definition of the OCPC). On the OCPC with $\mathcal{O}(n \cdot \frac{\log n}{\log\log n})$ processors random permutations of n items can be generated in $\mathcal{O}(\log\log n)$ time, so that the variation distance from the uniform probability distribution over $\mathbb{S}_n$ is $\mathcal{O}(\frac{1}{n})$. Similarly, one may implement distributed mixing on the QRQW PRAM (see [15] for a definition). In this case, the runtime would be $\mathcal{O}(\sqrt{\log n})$, since choosing the partners for exchange requires now considerably more time than for the OCPC. The previous best QRQW algorithm for generating random permutations [16] runs in time $\Theta(\log n)$ (but is work optimal, what might be hard to obtain for sublogarithmic runtimes).

3.2 Work fairness. One of large sources of computational power is utilization of free processor cycles in sys-

tems consisting of many workstations. However, implementing it efficiently without significant degradations of main services is a challenging problem (see [25] for a general survey). Unpredictability of free resources, computational requirements of the jobs utilizing free cycles, their duration, etc., makes it extremely difficult to design efficient and fair protocols for load balancing, load sharing, and similar goals. The results obtained are far from being complete.

There is a large class of jobs that can be moved within the system without a large overhead. (Examples of this kind are brute–force search, simulated annealing, genetic algorithms, branch-and-bound, factoring large numbers, breaking cryptographic ciphertexts, etc.) A simple approach that might be successful in practice for such jobs is to exchange them between the workstations at random. This provides a certain degree of utilization of free processor cycles. Also, this protocol is quite immune to such problems as poor service at certain locations at certain times. Furthermore, no machine is influenced by such a job for a longer time.

A simple intuitive approach might be to make each workstation pass its "movable" job to a randomly chosen location from time to time. However, then unavoidably some hot spots are generated, where many jobs arrive. Permuting jobs between workstations at random does not have this unwanted feature, but its implementation in a distributed system is non-obvious. However, distributed mixing is an efficient solution of this problem. Theorem 2.1 guarantees that with high probability after a short time the jobs are quite well allocated.

Our approach of using random matchings is related to the techniques from [13, 14]. However, our analysis uses Markov chain approach. Finally, we claim that many goals can be achieved by combining approaches of [13, 14] with distributed mixing.

3.3 Cryptographic defense against traffic analysis. Protecting messages against an eavesdropper has been a subject of intensive research. Tools preventing an eavesdropper from understanding or modifying a message without being detected have been developed. While it is known how to protect the contents of a message, it is a challenging problem how to hide that a message has been sent from one to another location, i.e. how to defend themselves against *traffic analysis*. This is a fundamental problem in many areas, where anonymity of communication is necessary.

Solving this problem is easy, when communication pattern is oblivious (like in a token ring). One strategy to make communication pattern oblivious is by introducing fake messages and randomness (see [18] for a software protection scheme of this kind). Another approach of hiding communication is to use *mixes* [8]. A mix is a node which obtains two messages encrypted with a public key of the mix, decodes them, encodes them with other keys and outputs to the next nodes. By properties of encryption, the eavesdropper cannot say where a given ciphertext has been forwarded after recoding inside the mix. The idea is that if we apply sufficiently many mixes and sufficiently many messages are sent, the eavesdropper has no chance to determine the destinations of the messages.

Rackoff and Simon [24] propose a protocol against traffic analysis related to mixes. They consider a fully connected network with n nodes where each node sends a message. The communication between the nodes is visible for the eavesdropper; moreover, the eavesdropper is able to conquer any subset of up to cn nodes, where $c < 1$ is a constant. In this protocol, if node A wants to send a message m to node B, then A chooses a random path $C_1, \ldots, C_\lambda$ leading from A to $B = C_\lambda$ (λ is a security parameter). Then m is encoded as $E_{k_1}(C_2, E_{k_2}(C_3, E_{k_4}(\ldots E_{k_\lambda}(m)\ldots)))$ and sent to node C_1 (k_i is the public key of node C_i). Then C_1 decodes the message with its private key, learns C_2 and sends the rest of the message, i.e. $E_{k_{C_2}}(C_3, E_{k_4}(\ldots E_{k_{C_\lambda}}(m)\ldots))$, to C_2. This is continued until the message reaches B.

Rackoff and Simon claim that for λ, which is polylogarithmic in n, the adversary cannot analyze the traffic. The conjecture is supported by a proof for a slightly different process, in which messages must remain inside certain (random) groups at certain phases of the protocol (for details see [24, Section 3.2]). The proof is based on convergence of the stochastic process defined in Theorem 2.3. We reduce the upper bound on the mixing time of this process from a polynomial in $\log n$ to $O(\log n)$. Due to a high degree (two-digit [26]) of the mentioned polynomial, the size of security parameter λ was previously too large and prevented practical applications [26]. Now we know that already $\lambda = O(\log^2 n)$ provides good security.

3.4 Communication services. Generating random permutations in hardware is an important problem due to its practical applications. For instance, such a hardware enclosed in a tamper-proof box would be a reliable source of randomness for many cryptographic protocols. Random and dynamic allocation of resources (such as communication channels) seems to be a very useful technique in telecommunication technology. (For example, frequent changing frequencies in GSM networks is used to achieve more reliable transmission and protection against eavesdropping.)

Permutation networks can be used to generate permutations at random by setting the switches at random and sending the packets through these switches. Though each permutation may be generated in this way, the classical permutation networks, like Beneš network, have some undesirable properties. Namely, there are permutations that correspond to very few settings of switches, and there are privileged ones that may be obtained in relatively many ways. It was already known that butterflies have some good "mixing" properties [22], but these properties are different from these we discuss here.

An elegant and simple idea to cope with the problem would be to pipeline several permutation networks. This corresponds to taking product of permutations generated by single networks, and the idea is that each permutation may be generated by composing "rare" and "frequent" permutations so eventually the differences between frequencies are significantly reduced. Experiments look good, but they do provide no reliable information due to the size of the set of all permutations.

Theorem 2.2 provides a simple switching network that *provably* generates permutations almost uniformly at random. We feel that some elements of the current construction as well as the upper bound on the mixing time, are due to present proof techniques and may be considerably improved in the future.

3.5 Fault resistant systems. Highly reliable systems are usually composed from independent components performing the same tasks. If one of the components fails, then the other components take over and provide regular service. This works as long as the faults are random and independent. However, an adversary may easily make the system brake down by damaging the components performing a single chosen task in the system.

A simple defense against faults caused by an adversary is to provide dynamic and random allocation of components to independent processors. If an adversary has no knowledge about allocation of tasks of the system, all he can do is to damage arbitrary processors. For example, assume that k different tasks are implemented on k^2 processors, each task by k processors. If an adversary knows the allocation, then he may damage exactly those k processors that perform a single function. However, if allocation of processors to tasks is random, the adversary has to damage much more processors. For example, damaging $k^2/2$ processors gives the probability less than $k/2^k$ that at least one task is not serviced.

Providing random and dynamic allocation of tasks to processors can be easily achieved by distributed mixing. Moreover, it is easy to execute the protocol on the system with damaged processors. Even if a constant fraction of n processors is damaged, then the bound from Theorem 2.1 still holds, i.e. after $\mathcal{O}(\log n)$ steps the allocation of tasks, which survived damaging the processors, is random.

4 Technical contribution.

Let $\mathfrak{M} = (\mathcal{Y}_t)_{t\in\mathbb{N}}$ be a discrete-time (possibly time-dependent) Markov chain with a finite state space $\mathfrak{Y}$ and the unique stationary distribution μ. Let $\mathcal{L}_Y(\mathcal{Y}_t)$ denote the probability distribution of $\mathcal{Y}_t$ given that $\mathcal{Y}_0 = Y$. We are interested in studying Markov chains for which the variation distance between $\mathcal{L}_Y(\mathcal{Y}_t)$ and μ tends quickly to zero, independently of Y. The standard measure of the convergence is the *mixing time*, denoted by $\tau_{\mathfrak{M}}(\varepsilon)$, which is defined as

$$\tau_{\mathfrak{M}}(\varepsilon) = \min\{T \in \mathbb{N} : \|\mathcal{L}_Y(\mathcal{Y}_t) - \mu\| \le \varepsilon \text{ for each } Y \in \mathfrak{Y} \text{ and each } t \ge T\} .$$

In this paper, we are interested in *very rapid mixing*, i.e., for $\varepsilon = 1/\mathrm{poly}(n)$ the mixing time is upper bounded by a (low degree) polynomial in $\log n$.

Coupling. A *coupling* [1, 10] for a Markov chain $\mathfrak{M} = (\mathcal{Y}_t)_{t\in\mathbb{N}}$ is a stochastic process $(Y_t, Y_t^*)_{t\in\mathbb{N}}$ on $\mathfrak{Y} \times \mathfrak{Y}$ such that each of $(Y_t)_{t\in\mathbb{N}}$, $(Y_t^*)_{t\in\mathbb{N}}$, considered independently, is a faithful copy of $(\mathcal{Y}_t)_{t\in\mathbb{N}}$ (i.e., $\mathcal{L}_Y(\mathcal{Y}_t) = \mathcal{L}_Y(Y_t) = \mathcal{L}_Y(Y_t^*)$ for each $Y \in \mathfrak{Y}$). The key result on coupling, the so-called *coupling inequality* (see, e.g., [1, Lemma 3.6]), states that the variation distance between $\mathcal{L}(\mathcal{Y}_t)$ and μ is bounded above by the probability that $Y_t \neq Y_t^*$ for the worst choice of initial states Y_0 and Y_0^*.

One difficulty in applying coupling lies in analyzing process $(Y_t, Y_t^*)_{t\in\mathbb{N}}$ on the whole space $\mathfrak{Y} \times \mathfrak{Y}$. The *path coupling* method of Bubley and Dyer [5] (see also [6, 7, 12]) allows to consider a coupling only for a subset of $\mathfrak{Y} \times \mathfrak{Y}$.

Path Coupling. Let Δ be a metric defined on $\mathfrak{Y} \times \mathfrak{Y}$ which takes values in $\{0, \ldots, D\}$. Let $\Gamma = \{(Y, Y^*) \in \mathfrak{Y} \times \mathfrak{Y} : \Delta(Y, Y^*) = 1\}$. ¿From now on we assume that for all $(Y, Y^*) \in \mathfrak{Y} \times \mathfrak{Y}$, if $\Delta(Y, Y^*) = r$, then there exists a sequence $Y = \Lambda_0, \Lambda_1, \ldots, \Lambda_r = Y^*$, with $(\Lambda_i, \Lambda_{i+1}) \in \Gamma$ for $0 \le i < r$.

Bubley and Dyer [5] showed that it is enough to consider a coupling only for $(Y, Y^*) \in \Gamma$.

LEMMA 4.1. (PATH COUPLING LEMMA [5], a simplified version) *Assume that there exists a coupling* $(Y_t, Y_t^*)_{t\in\mathbb{N}}$ *for* $\mathfrak{M}$ *such that for some* $\beta < 1$ *we have* $\mathbf{E}[\Delta(Y_{t+1}, Y_{t+1}^*)] \le \beta$ *for all* $(Y_t, Y_t^*) \in \Gamma$ *and* $t \in \mathbb{N}$. *Then* $\tau_{\mathfrak{M}}(\varepsilon) \le \left\lceil \frac{\ln(D\cdot\varepsilon^{-1})}{\ln(\beta^{-1})} \right\rceil$.

Delayed Path Coupling. In our analysis, we use a simple but powerful extension of the path coupling

method. In path coupling, the key step is to design a coupling $(Y_t, Y_t^*) \mapsto (Y_{t+1}, Y_{t+1}^*)$ defined only on pairs $(Y_t, Y_t^*) \in \Gamma$ such that $\mathbf{E}[\Delta(Y_{t+1}, Y_{t+1}^*)] < 1$ (or $\mathbf{E}[\Delta(Y_{t+1}, Y_{t+1}^*)] = 1$ and $\mathbf{Pr}[\Delta(Y_{t+1}, Y_{t+1}^*) \neq 1] < 1$, see [5]). However, we may have $\mathbf{Pr}[\Delta(Y_{t+1}, Y_{t+1}^*) \geq 1] = 1$ for certain $(Y_t, Y_t^*) \in \Gamma$. In this case, path coupling cannot be used directly. We may elude this problem by *delaying* the decision about coupling of the two processes until enough many steps are made.

Let $\mathfrak{M} = (\mathcal{Y}_t)_{t \in \mathbb{N}}$ be a Markov chain on $\mathfrak{Y}$. Let δ be a positive integer and let us consider the Markov chain $\mathfrak{M}_\delta = (\mathcal{Y}_t^\delta)_{t \in \mathbb{N}}$ on $\mathfrak{Y}$ such that $\mathcal{L}_Y(\mathcal{Y}_t^\delta) = \mathcal{L}_Y(\mathcal{Y}_{\delta \cdot t})$ for every $Y \in \mathfrak{Y}$. Obviously, if $\mathfrak{M}$ is ergodic, then so is $\mathfrak{M}_\delta$. Further, the stationary distributions of $\mathfrak{M}$ and $\mathfrak{M}_\delta$ are the same and $\tau_{\mathfrak{M}}(\varepsilon) \leq \delta \cdot \tau_{\mathfrak{M}_\delta}(\varepsilon)$ for every ε. Therefore Path Coupling Lemma applied to $\mathfrak{M}_\delta$ yields the following lemma:

LEMMA 4.2. (DELAYED PATH COUPLING LEMMA) *Let δ be a positive integer. Let $(Y_t, Y_t^*)_{t \in \mathbb{N}}$ be a coupling for $\mathfrak{M}$ such that for every $\tau \in \mathbb{N}$ and every $(Y_{\tau\delta}, Y_{\tau\delta}^*) \in \Gamma$ we get $\mathbf{E}[\Delta(Y_{(\tau+1)\delta}, Y_{(\tau+1)\delta}^*)] \leq \beta$ for some real $\beta < 1$. Then*

$$\tau_{\mathfrak{M}}(\varepsilon) \leq \delta \cdot \left\lceil \frac{\ln(D \cdot \varepsilon^{-1})}{\ln(\beta^{-1})} \right\rceil . \tag{4.1}$$

In particular, if $\mathbf{E}[\Delta(Y_{(\tau+1)\delta}, Y_{(\tau+1)\delta}^*)] \leq \frac{1}{n}$ for some $\delta = \mathcal{O}(\log n)$ and $D = \mathcal{O}(n)$, then

$$\tau_{\mathfrak{M}}(\tfrac{1}{n}) \leq \mathcal{O}(\log n) . \tag{4.2}$$

Let us remark that if Markov chain $\mathfrak{M}$ is time-independent, then in order to use Delayed Path Coupling Lemma it is enough to provide a bound for $\mathbf{E}[\Delta(Y_{(\tau+1)\delta}, Y_{(\tau+1)\delta}^*)]$ only for $\tau = 0$.

Although delayed path coupling is a simple refinement of the path coupling technique, it yields a very different approach to estimate the mixing times. It is worth to emphasize that delayed path coupling may lead to non-Markovian couplings (since a Markovian coupling for $\mathfrak{M}_\delta$ does not have to be Markovian for $\mathfrak{M}$).

Delayed path coupling can be used in a relatively simple way to prove Theorem 2.2 (to appear in the full version of the paper). More complicated delayed path coupling arguments are used in the proof of Theorem 2.1 (see Section 5), where a non-Markovian coupling is constructed to bound the mixing time of the matching exchange protocol. Finally, more sophisticated arguments are incorporated in our proof of Theorem 2.3 (to appear in the full version of the paper), where we show the *existence* of a non-Markovian coupling.

5 Analysis of the matching exchange process – an outline.

Let $\mathfrak{M} = (\Pi_t)_{t \in \mathbb{N}}$ be a Markov chain with the state space $\mathbb{S}_n$ and transitions made according to the transition rules of the matching exchange protocol. We define the distance metric between $\pi, \pi^* \in \mathbb{S}_n$, denoted by $\Delta(\pi, \pi^*)$, as the minimum number of transpositions required to transform π into π^*. We apply the delayed path coupling method. We pick sufficiently large $\tau = \Theta(\log n)$ and define a coupling $(\pi_t, \pi_t^*)_{t=0}^{\tau}$ for $\mathfrak{M}$ with $\Delta(\pi_0, \pi_0^*) = 1$. Our aim is to show that $\mathbf{E}[\Delta(\pi_\tau, \pi_\tau^*)] \leq \frac{1}{n}$, which implies Theorem 2.1 by inequality (4.2).

In order to define the coupling, we first pick τ matchings $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ in G and active matchings $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ generated by $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ according to the matching exchange protocol. They are the matchings used by the first process to define $\pi_1, \ldots, \pi_\tau$. Then, for $(\pi_t^*)_{t=1}^{\tau}$, we define matchings $\mathbf{M}_1^*, \ldots, \mathbf{M}_\tau^*$ in G and active matchings $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$ generated by $\mathbf{M}_1^*, \ldots, \mathbf{M}_\tau^*$, as discussed below.

If edges $\{i,k\}$ and $\{j,l\}$ are in a matching M, then let $M[i \leftrightarrow j]$ denote the matching $(M - \{\{i,k\},\{j,l\}\}) \cup \{\{i,l\},\{j,k\}\}$. Let us fix π_0 and π_0^* that differ only on positions i_0 and j_0, i.e., $\pi_0^*(s) = \pi_0(s)$ if and only if $s \neq (\pi_0)^{-1}(i_0), (\pi_0)^{-1}(j_0)$.

Notice first that if $\{i_0, j_0\} \in \mathbf{M}_1$, then we may obtain $\pi_1 = \pi_1^*$. Indeed, we set $\mathbf{M}_1^* = \mathbf{M}_1$ and $\tilde{\mathbf{M}}_1^* = \tilde{\mathbf{M}}_1 \cup \{\{i_0, j_0\}\}$ if $\{i_0, j_0\} \notin \tilde{\mathbf{M}}_1$ and $\tilde{\mathbf{M}}_1^* = \tilde{\mathbf{M}}_1 - \{\{i_0, j_0\}\}$ otherwise. Then obviously $\pi_1 = \pi_1^*$. However, the probability that $\{i_0, j_0\} \in \mathbf{M}_1$ is $\Theta(\frac{1}{n})$ only![1] Our main observation is that we can proceed further on with the next matchings to improve the success probability. Suppose that $\{i_0, k\}, \{j_0, l\} \in \tilde{\mathbf{M}}_1$ for some k, l. Now there are two possible couplings for $\tilde{\mathbf{M}}_1^*$ between which we may choose. If we see, for example, that $\{i_0, j_0\} \in \mathbf{M}_2$, then we set $\tilde{\mathbf{M}}_1^* = \tilde{\mathbf{M}}_1[i_0 \leftrightarrow j_0]$ (see Fig. 1). Then π_1 and π_1^* differ only at positions i_0 and j_0, and the arguments given above can be used to define $\tilde{\mathbf{M}}_2^*$ such that $\pi_2 = \pi_2^*$. On the other hand, if $\{k, l\} \in \mathbf{M}_2$, then we set $\tilde{\mathbf{M}}_1^* = \tilde{\mathbf{M}}_1$. Then π_1 and π_1^* differ only at positions k and l and we may set $\tilde{\mathbf{M}}_2^*$ such that $\pi_2 = \pi_2^*$. Thus, informally, we can set π_1^* and π_2^* such that the probability of getting $\pi_2 = \pi_2^*$ is roughly twice as big as the probability that $\pi_1 = \pi_1^*$. We can continue these arguments further, and show that for some $T = \Theta(\log n)$ and for almost every $\mathbf{M}_1, \ldots, \mathbf{M}_T$ and $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_T$ with $\tilde{\mathbf{M}}_t \subseteq \mathbf{M}_t$, we can use the method described above to construct "good" sequences $\mathbf{M}_1^*, \ldots, \mathbf{M}_T^*$ and $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_T^*$ with $\tilde{\mathbf{M}}_t^* \subseteq \mathbf{M}_t^*$, such that $\pi_T = \pi_T^*$.

[1] This is the main reason why there was no $\mathcal{O}(n \log n)$ coupling-based proof for the random transpositions process [1, 10].

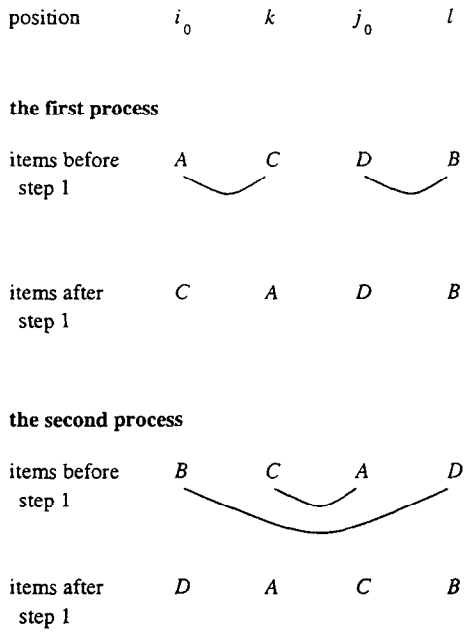

Figure 1: Preserving the place of difference between two processes of coupling at step 1

One non-obvious issue has not been considered yet: we must construct π_t^* that is a faithful copy of $\mathfrak{M}$. For this purpose we refine the construction of matchings $\mathbf{M}_1^*, \ldots, \mathbf{M}_\tau^*$ and $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$. Our main argument is roughly as follows. Observe that given $k_1, \ldots, k_\tau$, the probability that the matching exchange protocol picks matchings $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ in G with $|\mathbf{M}_i| = k_i$, and active matchings $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ with $\tilde{\mathbf{M}}_t \subseteq \mathbf{M}_t$, is the same for every pair of sequences $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ and $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$. Therefore our aim is to define a *permutation* $\mathfrak{S}$ such that if $(\pi_t)_{t=0}^\tau$ is defined by active matchings $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ generated by matchings $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ in G and $(\pi_t^*)_{t=0}^\tau$ is defined by matchings given by $\mathfrak{S}(\mathbf{M}_1, \ldots, \mathbf{M}_\tau; \tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau)$, then $\mathbf{E}[\Delta(\pi_\tau, \pi_\tau^*)] \leq \frac{1}{n}$. Since $\mathfrak{S}$ is a permutation, the marginal distribution of matchings chosen for the second process is the same as for the first process. Hence such $(\pi_t, \pi_t^*)_{t=0}^\tau$ would be a proper coupling and Delayed Path Coupling Lemma (inequality (4.2)) would imply Theorem 2.1.

Let us finally notice that since $\mathfrak{S}$ is defined for the whole sequences $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ and $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$, our coupling is non-Markovian.

6 Details of the proof of Theorem 2.1.

6.1 Good pairs. For active matchings $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$, we define *good pairs* at a given step and *traces* corresponding to the good pairs. Each good pair is a pair of integers $\{i, j\}$ for $i, j \in \{1, \ldots, n\}$. Each trace is a set of elements of the form $(t; k \leftrightarrow l)$, where t is an integer, $0 < t < \tau$, and $k, l \in \{1, \ldots, n\}$. A trace provides information necessary to define π_t^* so that a difference between π_t^* and π_t occurs at the positions given by the good pair. We define the good pairs and their traces by induction on t. The only good pair at step 0 is $\{i_0, j_0\}$ (we remind that π_0 and π_0^* differ on positions i_0 and j_0) and its trace is empty. Now let GP_{t-1} be the set of good pairs at step $t-1$ and $I_{t-1} = \{k : \exists l\ \{k, l\} \in GP_{t-1}\}$. We define GP_t by considering each pair in GP_{t-1} one by one. Let $\{i, j\} \in GP_{t-1}$ and let P be the trace for $\{i, j\}$ at step $t-1$:

- if neither of i and j is incident to any edge in $\tilde{\mathbf{M}}_t$, then $\{i, j\} \in GP_t$ and P is the trace for $\{i, j\}$ at step t.
- if $\{i, k\} \in \tilde{\mathbf{M}}_t$ and j is not incident to any edge in $\tilde{\mathbf{M}}_t$, then $\{k, j\} \in GP_t$ and P is the trace for $\{k, j\}$ at step t.
- if $\{i, k\}, \{j, l\} \in \tilde{\mathbf{M}}_t$, then
  - if $k, l \notin I_{t-1}$, then
    - $\{k, l\} \in GP_t$ and P is the trace for $\{k, l\}$ at step t, and
    - $\{i, j\} \in GP_t$ and $P \cup \{(t; i \leftrightarrow j)\}$ is the trace for $\{i, j\}$ at step t.
  - otherwise $\{k, l\} \in GP_t$ and P is the trace for $\{k, l\}$ at step t.

It is easy to see that our construction ensures that each $i \in \{1, \ldots, n\}$ may be an element of at most one good pair at step t, and therefore $|GP_t| \leq n/2$ for every $1 \leq t \leq \tau$. We show now that the number of good pairs will be almost surely $\Theta(n)$ after $\mathcal{O}(\log n)$ steps.

CLAIM 6.1. *There is a $T = \Theta(\log n)$ such that the probability (over the choices of $k_1, \ldots, k_\tau$, of matchings $\mathbf{M}_1, \ldots, \mathbf{M}_T$ in G, and of active matchings $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_T$) that $|GP_T| = o(n)$ is at most $\frac{1}{n^2}$.*

Proof. Let a positive constant c_1 fulfill $\mathbf{E}[k_t] \geq c_1 n$. Then $\mathbf{E}[|\tilde{\mathbf{M}}_t|] \geq c_1 n/2$ for every $t \in \mathbb{N}$. Therefore, Markov's inequality implies that $\mathbf{Pr}[|\tilde{\mathbf{M}}_t| \leq c_1 n/4] \leq 1 - c_2$ for $c_2 = c_1/(2 - c_1)$. Let X_t be the indicator random variable for the event $|\tilde{\mathbf{M}}_t| > c_1 n/4$. Hence $\mathbf{Pr}[X_t = 1] \geq c_2$ and Chernoff bound yields

$$\mathbf{Pr}[\textstyle\sum_{t=1}^T X_t \leq \frac{1}{2} c_2 T] \leq \frac{1}{2} n^{-2} \tag{6.3}$$

for $T \geq 24 \ln n / c_2$. Now we shall condition on the event $\sum_{t=1}^T X_t > \frac{1}{2} c_2 T$. Let $\mathfrak{P}$ be a matching exchange process that picks $\frac{1}{2} c_2 T$ active matchings, each of size at least $c_1 n/4$. Conditioned on the event $\sum_{t=1}^T X_t > \frac{1}{2} c_2 T$, the number of good pairs in GP_T stochastically dominates the number of good pairs in $\mathfrak{P}$. Let Y_t

be the number of good pairs at step t in process $\mathfrak{P}$, $0 \leq t \leq \frac{1}{2}c_2T$. Clearly $Y_0 = 1$. Now, let $\{i,j\}$ be a good pair at step $t-1$ in $\mathfrak{P}$. This pair increases the number of good pairs at step t only if $\{i,k\}$, $\{j,l\}$ are the edges of the t-th active matching in $\mathfrak{P}$ for $k,l \notin I_{t-1}$. The probability of this event is lower bounded by a positive constant c_3, provided that $Y_{t-1} \leq n/4$. Therefore, if $Y_{t-1} \leq n/4$, then $\mathbf{E}[Y_t|Y_{t-1}] \geq Y_{t-1}(1+c_3)$. This clearly yields that there is a $T = \Theta(\log n)$ such that $\mathbf{Pr}[Y_T \leq n/4] \leq \frac{1}{2}n^{-2}$. We combine this inequality with (6.3) to complete the proof of Claim 6.1.

LEMMA 6.1. *There is a $T_0 = \Theta(\log n)$ such that the probability (over the choices of $k_1,\ldots,k_\tau$, of matchings $\mathbf{M}_1,\ldots,\mathbf{M}_{T_0}$ in G, and of active matchings $\tilde{\mathbf{M}}_1,\ldots,\tilde{\mathbf{M}}_{T_0}$) that $GP_{t-1} \cap \mathbf{M}_t = \emptyset$ for all $t \leq T_0$ is at most $\frac{1}{n}$.*

Proof. We shall condition on the event that $|GP_{\lfloor T_0/2 \rfloor}| \geq c_4 n$ for some constant $c_4 > 0$. Further, similarly as in the proof of Claim 6.1, we shall condition on the event that at least $c_5 T_0$ matchings among $\mathbf{M}_{1+\lfloor T_0/2 \rfloor},\ldots,\mathbf{M}_{T_0}$ are of size $c_6 n$, for some positive constants c_5 and c_6. By the arguments used in the proof of Claim 6.1, we know that any of these events does not hold with probability at most $\frac{1}{2}n^{-2}$ for an appropriate choice of c_5 and c_6.

Observe that for each step t all good pairs are disjoint. Notice also that the matchings generated by $\mathfrak{M}$ are independent of the positions of the good pairs. Therefore we can model our problem as follows:
Let H be a complete graph with n vertices and let AM be an arbitrary matching in H of size at least $c_4 n$ (i.e., each matching edge corresponds to a good pair). Pick $c_5 T$ random matchings $RM_1,\ldots,RM_{c_5 T}$ in H, each of size at least $c_6 n$. Now, the probability that none of the matchings $RM_1,\ldots,RM_{c_5 T_0}$ hits an edge from AM is an upper bound for the probability that $\bigcup_{t=1}^{T_0}(GP_{t-1} \cap \mathbf{M}_t) = \emptyset$.

Since it is easy to see that $\mathbf{Pr}[AM \cap RM_t = \emptyset] \leq c_7$ for some constant $c_7 < 1$, we obtain

$$\mathbf{Pr}[AM \cap (RM_1 \cup \cdots \cup RM_{c_5 T_0}) = \emptyset] \leq (c_7)^{c_5 T_0} .$$

Therefore we can complete the proof of Lemma 6.1 by setting $T_0 \geq \frac{2 \cdot \ln n}{c_5 \cdot \ln(1/c_7)}$.

6.2 Critical active matchings. For a given good pair and its trace P at step T for active matchings $\tilde{\mathbf{M}}_1,\ldots,\tilde{\mathbf{M}}_T$, we define the *critical active matchings* $\tilde{\mathbf{M}}_1^P,\ldots,\tilde{\mathbf{M}}_T^P$ such that for each t, $1 \leq t \leq T$:

- if $(t; k \leftrightarrow l) \in P$, then $\tilde{\mathbf{M}}_t^P = \tilde{\mathbf{M}}_t[k \leftrightarrow l]$;
- otherwise $\tilde{\mathbf{M}}_t^P = \tilde{\mathbf{M}}_t$.

The following claim states the key property of good pairs, their traces, and critical active matchings.

CLAIM 6.2. *Let $\{p,q\}$ be a good pair with the trace P at step t for active matchings $\tilde{\mathbf{M}}_1,\ldots,\tilde{\mathbf{M}}_t$ and let $\tilde{\mathbf{M}}_1^P,\ldots,\tilde{\mathbf{M}}_t^P$ be the corresponding critical active matchings. Let $(\pi_t)_{t=0}^{\tau}$ and $(\pi_t^*)_{t=0}^{\tau}$ be the permutations defined by $\tilde{\mathbf{M}}_1,\ldots,\tilde{\mathbf{M}}_\tau$ and by $\tilde{\mathbf{M}}_1^P,\ldots,\tilde{\mathbf{M}}_\tau^P$, respectively. Then π_t and π_t^* differ only on positions p and q (i.e., $\pi_t(s) = \pi_t^*(s)$ for all $s \neq (\pi_t)^{-1}(p), (\pi_t)^{-1}(q)$).*

Proof. The proof is by induction on t. The claim clearly holds for $t = 0$. Now we assume that it is true for $t-1$ and we show that it is valid for t.

If $\{p,q\}$ is a good pair at step t, then it has been created by a good pair $\{i,j\}$ with a trace P^* at step $t-1$. Notice that in that case $\pi_{t-1}(s) = \pi_{t-1}^*(s)$ for all $s \neq (\pi_{t-1})^{-1}(i), (\pi_{t-1})^{-1}(j)$, by the inductive hypothesis. We consider the following cases:

- If $i = p$, $j = q$, and neither of i and j is incident to any edge in $\tilde{\mathbf{M}}_t$, then $P = P^*$, and hence $\tilde{\mathbf{M}}_t^P = \tilde{\mathbf{M}}_t$. This yields $\pi_t(s) = \pi_t^*(s)$ for all $s \neq (\pi_{t-1})^{-1}(i), (\pi_{t-1})^{-1}(j)$, and the claim follows by the fact that $(\pi_t)^{-1}(p) = (\pi_{t-1})^{-1}(i)$ and $(\pi_t)^{-1}(q) = (\pi_{t-1})^{-1}(j)$.
- If $j = q$, $\{i,p\} \in \tilde{\mathbf{M}}_t$, and j is not incident to any edge in $\tilde{\mathbf{M}}_t$, then $P = P^*$, and hence $\tilde{\mathbf{M}}_t^P = \tilde{\mathbf{M}}_t$. Therefore $\pi_t(s) = \pi_t^*(s)$ for all $s \neq (\pi_{t-1})^{-1}(i), (\pi_{t-1})^{-1}(q)$. Now observe that $(\pi_t)^{-1}(p) = (\pi_{t-1})^{-1}(i)$ and $(\pi_t)^{-1}(q) = (\pi_{t-1})^{-1}(q)$, and hence the claim holds for t.
- If $\{i,j\} \cap \{p,q\} = \emptyset$ and $\{i,p\}, \{j,q\} \in \tilde{\mathbf{M}}_t$, then $P = P^*$ and $\tilde{\mathbf{M}}_t^P = \mathbf{M}_t$. Therefore, similarly as above, $\pi_t(s) = \pi_t^*(s)$ for all $s \neq (\pi_{t-1})^{-1}(i), (\pi_{t-1})^{-1}(j)$. Now the claim follows from the fact that $(\pi_t)^{-1}(p) = (\pi_{t-1})^{-1}(i)$ and $(\pi_t)^{-1}(q) = (\pi_{t-1})^{-1}(j)$.
- If $i = p$, $j = q$, and $\{i,k\}, \{j,l\} \in \tilde{\mathbf{M}}_t$, then $k,l \notin I_{t-1}$ and $P = P^* \cup \{(t; i \leftrightarrow j)\}$. Since $\tilde{\mathbf{M}}_t^P = \tilde{\mathbf{M}}_t[i \leftrightarrow j]$, we have $\pi_t(s) = \pi_t^*(s)$ for $s \neq (\pi_{t-1})^{-1}(i), (\pi_{t-1})^{-1}(j), (\pi_{t-1})^{-1}(k), (\pi_{t-1})^{-1}(l)$. On the other hand, since $\{i,k\}, \{j,l\} \in \tilde{\mathbf{M}}_t$ and $\{i,l\}, \{j,k\} \in \tilde{\mathbf{M}}_t^P$, we have $\pi_t((\pi_{t-1})^{-1}(i)) = k = \pi_t^*((\pi_{t-1}^*)^{-1}(j)) = \pi_t^*((\pi_{t-1})^{-1}(i))$. Similarly, $\pi_t((\pi_{t-1})^{-1}(j)) = l = \pi_t^*((\pi_{t-1}^*)^{-1}(i)) = \pi_t^*((\pi_{t-1})^{-1}(j))$. Therefore $\pi_t(s) = \pi_t^*(s)$ for $s = (\pi_t)^{-1}(k)$ and $s = (\pi_t)^{-1}(l)$, which yields the claim.

6.3 Sibling matchings. In this subsection we define permutation $\mathfrak{S}$. Now let T_0 be as in Lemma 6.1; we require that τ is set so that $\tau \geq T_0$. First, we proceed conditioned on the event that $\bigcup_{t=1}^{T_0}(GP_{t-1} \cap \mathbf{M}_t) \neq \emptyset$.

Let $\mathfrak{T} \leq T_0$ be the smallest t such that there is a good pair at step $t-1$ that is in $\mathbf{M}_t$ (i.e., $GP_{t-1} \cap \mathbf{M}_t \neq \emptyset$); we call $\mathfrak{T}$ the *meeting time* for $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$. Let $\{i,j\}$ be the *lexicographically first* good pair at step $\mathfrak{T}-1$ that is in $\mathbf{M}_\mathfrak{T}$ and let P be its trace. We call pair $\{i,j\}$ the *terminal pair* for $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$.

We define now *sibling matchings* $\mathbf{M}_1^*, \ldots, \mathbf{M}_\tau^*$ and *sibling active matchings* $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$ for matchings $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ and $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$. For every t, $1 \leq t < \mathfrak{T}$, we set $\tilde{\mathbf{M}}_t^*$ as follows:

- if $(t; k \leftrightarrow l) \in P$, then $\mathbf{M}_t^* = \mathbf{M}_t[k \leftrightarrow l]$ and $\tilde{\mathbf{M}}_t^* = \tilde{\mathbf{M}}_t[k \leftrightarrow l]$;
- otherwise $\mathbf{M}_t^* = \mathbf{M}_t$ and $\tilde{\mathbf{M}}_t^* = \tilde{\mathbf{M}}_t$.

We also define $\mathbf{M}_\mathfrak{T}^* = \mathbf{M}_\mathfrak{T}$ and

$$\tilde{\mathbf{M}}_\mathfrak{T}^* = \begin{cases} \tilde{\mathbf{M}}_\mathfrak{T} - \{\{i,j\}\} & \text{if } \{i,j\} \in \tilde{\mathbf{M}}_\mathfrak{T} \\ \tilde{\mathbf{M}}_\mathfrak{T} \cup \{\{i,j\}\} & \text{if } \{i,j\} \notin \tilde{\mathbf{M}}_\mathfrak{T} \end{cases},$$

as well as $\mathbf{M}_t^* = \mathbf{M}_t$ and $\tilde{\mathbf{M}}_t^* = \tilde{\mathbf{M}}_t$ for all $\mathfrak{T} < t \leq \tau$. The sequence of sibling matchings $\mathbf{M}_1^*, \ldots, \mathbf{M}_\tau^*$ and *sibling active matchings* $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$ defined for $\mathbf{M}_1, \ldots, \mathbf{M}_\tau, \tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ will be denoted by $\mathfrak{S}(\mathbf{M}_1, \ldots, \mathbf{M}_\tau, \tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau)$.

Our definition of the sibling sequence yields the following two key claims.

CLAIM 6.3. *Let* $\bigcup_{t=1}^{\tau}(GP_{t-1} \cap \mathbf{M}_t) \neq \emptyset$ *and let* $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$ *be the sibling active matchings for matchings* $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ *and* $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$. *If* $(\pi_t)_{t=0}^{\tau}$ *and* $(\pi_t^*)_{t=0}^{\tau}$ *are defined by* $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ *and* $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$, *respectively, then* $\pi_\tau = \pi_\tau^*$.

Proof. Let $\mathfrak{T}$ be the meeting time and $\{i,j\}$ be the terminal pair for $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$. Then $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_{\mathfrak{T}-1}^*$, are the critical matchings for $\{i,j\}$ at time $\mathfrak{T}-1$. Therefore Claim 6.2 implies that $\pi_{\mathfrak{T}-1}$ and $\pi_{\mathfrak{T}-1}^*$ differ only on positions i and j. Now observe that $\tilde{\mathbf{M}}_\mathfrak{T}$ and $\tilde{\mathbf{M}}_\mathfrak{T}^*$ differ only on edge $\{i,j\}$ and that exactly one of $\tilde{\mathbf{M}}_\mathfrak{T}$ and $\tilde{\mathbf{M}}_\mathfrak{T}^*$ contains edge $\{i,j\}$. Therefore we obtain $\pi_\mathfrak{T} = \pi_\mathfrak{T}^*$. Now the claim follows from the fact that $\tilde{\mathbf{M}}_t = \tilde{\mathbf{M}}_t^*$ for every $\mathfrak{T}+1 \leq t \leq \tau$.

Claim 6.3 guarantees that the processes under consideration couple as desired. The next claim guarantees correctness of the construction in the sense that $\mathfrak{S}$ is a permutation and, as we shall see, the marginal probability distributions of the processes forming the coupling are as expected.

CLAIM 6.4. *Let* $\mathbf{M}_1^*, \ldots, \mathbf{M}_\tau^*$ *and* $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$ *be the sibling matchings and sibling active matchings for* $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ *and* $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$. *Then* $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ *and* $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ *are, respectively, the sibling matchings and sibling active matchings for* $\mathbf{M}_1^*, \ldots, \mathbf{M}_\tau^*$ *and* $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$.

Proof. We prove the claim by showing that the following property is satisfied for every $\{i,j\}$ and $1 \leq t \leq \tau$: $\{i,j\}$ is a good pair with a trace P at step t for $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ if and only if $\{i,j\}$ is a good pair with a trace P at step t for $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$.

We prove the property above by induction on t. Since the claim trivially holds for $t = 0$, we assume that it does hold for $t-1$ and will validate it for t.

By our construction we know that $\tilde{\mathbf{M}}_t^*$ equals either $\tilde{\mathbf{M}}_t$ or $\tilde{\mathbf{M}}_t[p \leftrightarrow q]$ for some good pair $\{p,q\}$ at step $t-1$ for $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ with $\{p,r\}, \{q,s\} \in \tilde{\mathbf{M}}_t$ for some $r, s \notin I_{t-1}$. Hence by the inductive hypothesis, $\{p,q\}$ is a good pair at step $t-1$ also for $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$. Clearly, if $\tilde{\mathbf{M}}_t^* = \tilde{\mathbf{M}}_t$, then the inductive hypothesis ensures that the property holds at step t. Thus let us suppose that $\tilde{\mathbf{M}}_t^* = \tilde{\mathbf{M}}_t[p \leftrightarrow q]$. Each good pair at step t is created by a good pair at step $t-1$. Let $\{k,l\}$ be a good pair at step $t-1$ (for both $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ and $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$). If $\{k,l\} \neq \{p,q\}$, then $\{k,l\}$ will create the same good pair(s) with the same traces at step t for $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ and $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$, since any edges incident to k and l are the same in $\tilde{\mathbf{M}}_t$ and $\tilde{\mathbf{M}}_t^*$. Now let us consider the case $\{k,l\} = \{p,q\}$ and let P be the trace for $\{p,q\}$ at step $t-1$ (for both $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ and $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$). Then there are some $r, s \notin I_{t-1}$ such that $\{p,r\}, \{q,s\} \in \tilde{\mathbf{M}}_t$ and $\{p,s\}, \{q,r\} \in \tilde{\mathbf{M}}_t^*$. Therefore $\{p,q\}$ is a good pair with trace $P \cup \{(t; p \leftrightarrow q)\}$ at step t for both $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ and $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$. Similarly, $\{r,s\}$ is a good pair with trace P for both $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ and $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$.

Once we know that the property holds, we obtain that the terminal pairs and their traces are the same for $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ and for $\tilde{\mathbf{M}}_1^*, \ldots, \tilde{\mathbf{M}}_\tau^*$. Therefore Claim 6.4 follows directly by the definitions of the sibling matchings.

In order to complete the definition of $\mathfrak{S}$ we consider now the remaining situation when $\bigcup_{t=1}^{T_0}(GP_{t-1} \cap \tilde{\mathbf{M}}_t) = \emptyset$. In that case we define the sibling matchings and the sibling active matchings for $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ and $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ as $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ and $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ itself. Obviously in this case $\Delta(\pi_\tau, \pi_\tau^*) = 1$.

This leads to the following final lemma.

LEMMA 6.2. *Let* π_0 *and* π_0^* *be two permutations that differ only in positions* i_0 *and* j_0. *Let* $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$ *be a sequence of matchings in* G *and let* $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$ *be any active matchings generated by* $\mathbf{M}_1, \ldots, \mathbf{M}_\tau$. *Let* $(\pi_t)_{t=0}^{\tau}$ *be defined by active matchings* $\tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau$. *If we define* $(\pi_t^*)_{t=0}^{\tau}$ *by* $\mathfrak{S}(\mathbf{M}_1, \ldots, \mathbf{M}_\tau, \tilde{\mathbf{M}}_1, \ldots, \tilde{\mathbf{M}}_\tau)$, *then* $(\pi_t, \pi_t^*)_{t=1}^{\tau}$ *is a coupling and* $\mathbf{E}[\Delta(\pi_\tau, \pi_\tau^*)] \leq \frac{1}{n}$.

Proof. Fix $k_1, \ldots, k_\tau$. Let us consider the function $\mathfrak{S}$ that maps matchings of prescribed sizes $k_1, \ldots, k_\tau$ and active matchings into their sibling matchings and sibling active matchings. For matchings such that $\bigcup_{t=1}^{T_0}(GP_{t-1} \cap \mathbf{M}_t) = \emptyset$ function $\mathfrak{S}$ is identity. Otherwise, by Claim 6.4, $\mathfrak{S}^2$ is identity. This implies that $\mathfrak{S}$ is a symmetric permutation over the set of matchings with prescribed sizes $k_1, \ldots, k_\tau$ and their active matchings. Since, as already seen at the beginning of Section 5, each such sequence of matchings has the same probability, $(\pi_t, \pi_t^*)_{t=0}^{\tau}$ is a valid coupling.

It remains to show that $\mathbf{E}[\Delta(\pi_\tau, \pi_\tau^*)] \leq \frac{1}{n}$. By Claim 6.3 we know that if there is some t, $1 \leq t \leq T_0$, such that $GP_{t-1} \cap \tilde{\mathbf{M}}_t$ is non-empty, then $\pi_\tau = \pi_\tau^*$. On the other hand, if $GP_{t-1} \cap \tilde{\mathbf{M}}_t = \emptyset$ for all $1 \leq t \leq T_0$, then $\Delta(\pi_\tau, \pi_\tau^*) = 1$. By Lemma 6.1, the latter event occurs with probability at most $\frac{1}{n}$, so $\mathbf{E}[\Delta(\pi_\tau, \pi_\tau^*)] \leq \frac{1}{n}$, as claimed.

Now we can complete the proof of Theorem 2.1 by combining Lemma 6.2 with Delayed Path Coupling Lemma.

References

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

[2] D. Aldous and P. Diaconis, *Shuffling cards and stopping times*, Amer. Math. Monthly, 93 (1986), 333–347.

[3] N. Alon, F. R. K. Chung, and R. L. Graham, *Routing permutations on graphs via matchings*, SIAM J. on Discrete Mathematics, 7 (1994), pp. 513–530.

[4] R. J. Anderson and G. L. Miller, *Optical communication for pointer based algorithms*, Tech. Report CRI 88-14, University of Southern California, 1988.

[5] R. Bubley and M. Dyer, *Path coupling: A technique for proving rapid mixing in Markov chains*, in Proc. 38th FOCS'97, pp. 223–231.

[6] R. Bubley and M. Dyer, *Faster random generation of linear extensions*, in Proc. 9th SODA'98, pp. 350–354.

[7] R. Bubley, M. Dyer, and C. Greenhill, *Beating the 2Δ bound for approximately counting colourings: A computer-assisted proof of rapid mixing*, in Proc. 9th SODA'98, pp. 355–363.

[8] D. Chaum, *Untraceable electronic mail, return addresses, and digital pseudonyms*, CACM, 24 (1985), pp. 84–88.

[9] A. Czumaj, P. Kanarek, M. Kutyłowski, and K. Loryś, *Fast generation of random permutations via network simulation*, Algorithmica, 21 (1998), pp. 2–20.

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

[11] M. Dyer, A. Frieze, and R. Kannan, *A random polynomial-time algorithm for approximating the volume of convex bodies*, JACM, 38 (1991), pp. 1–17.

[12] M. Dyer and C. Greenhill, *A genuinely polynomial-time algorithm for sampling two-rowed contingency tables*, in Proc. 25th ICALP'98, pp. 339–350.

[13] B. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajaraman, A. W. Richa, R. E. Tarjan, and D. Zuckerman, *Tight analyses of two local load balancing algorithms*, in Proc. 27th STOC'95, pp. 548–558.

[14] B. Ghosh and S. Muthukrishnan, *Dynamic load balancing by random matchings*, JCSS, 53 (1996), pp. 357–370.

[15] P. Gibbons, Y. Matias, V. Ramachandran, *Efficient low-contention parallel algorithms*, JCSS, 53 (1996), pp. 417–442.

[16] P. Gibbons, Y. Matias, V. Ramachandran, *The QRQW PRAM: accounting for contention in parallel algorithms*, in Proc. 5th SODA'94, pp. 638–648, to appear in SIAM J. Comput..

[17] L. A. Goldberg, M. Jerrum, T. Leighton, and S. Rao, *A doubly logarithmic communication algorithm for the completely connected optical communication parallel computer*, SIAM J. Comput., 26 (1997), pp. 1100–1119.

[18] O. Goldreich and R. Ostrovsky, *Software protection and simulation on oblivious RAMs*, JACM, 43 (1996), pp. 431–473.

[19] T. Hagerup, *Fast parallel generation of random permutations*, in Proc. 18th ICALP'91, pp. 405–416.

[20] M. Jerrum and A. Sinclair, *The Markov chain Monte Carlo method: An approach to approximate counting and integration*, in D. S. Hochbaum, editor, *Approximation Algorithms*, PWS Publishing Company, Boston, 1996, pp. 482-520.

[21] R. Kannan, *Markov chains and polynomial time algorithms*, in Proc. 35th FOCS'94, pp. 656–671.

[22] T. Leighton and C. G. Plaxton, *Hypercubic sorting networks*, SIAM J. Comput., 27 (1998), pp. 1–47.

[23] Y. Matias and U. Vishkin, *Converting high probability into nearly-constant time - with applications to parallel hashing*, in Proc. 23rd STOC'91, pp. 307–316.

[24] C. Rackoff and D. R. Simon, *Cryptographic defense against traffic analysis*, in Proc. 25th STOC'93, pp. 672–681.

[25] B. A. Shirazi, A. R. Hurson, K. M. Kavi, editors, *Scheduling and load balancing in parallel and distributed systems*, IEEE Computer Society Press, Los Alamitos, 1995.

[26] D. R. Simon, *private communication*, October 1997.

[27] A. Sinclair, *Algorithms for Random Generation and Counting: A Markov Chain Approach*, Progress in Theoretical Comp. Sci., Birkhäuser, Boston, 1993.

[28] M. Zito, I. Pu, M. Amos, and A. Gibbons, *RNC algorithms for the uniform generation of combinatorial structures*, in Proc. 7th SODA'96, pp. 429–437.