Computational and Applied Mathematics (2020) 39:244
https://doi.org/10.1007/s40314-020-01295-4

Efficient generation of random derangements with the expected distribution of cycle lengths

J. Ricardo G. Mendonça[1]

Received: 5 January 2020 / Revised: 5 January 2020 / Accepted: 7 August 2020 / Published online: 13 August 2020

Abstract

We show how to generate random derangements efficiently by two different techniques: random restricted transpositions and sequential importance sampling. The algorithm employing restricted transpositions can also be used to generate random fixed-point-free involutions only, a. k. a. random perfect matchings on the complete graph. Our data indicate that the algorithms generate random samples with the expected distribution of cycle lengths, which we derive, and for relatively small samples, which can actually be very large in absolute numbers, we argue that they generate samples indistinguishable from the uniform distribution. Both algorithms are simple to understand and implement and possess a performance comparable to or better than those of currently known methods. Simulations suggest that the mixing time of the algorithm based on random restricted transpositions (in the total variance distance with respect to the distribution of cycle lengths) is $O(n^a \log n^2)$ with $a \simeq \frac{1}{2}$ and n the length of the derangement. We prove that the sequential importance sampling algorithm generates random derangements in $O(n)$ time with probability $O(1/n)$ of failing.

Keywords Restricted permutation · Random transposition walk · Random perfect matching · Switch Markov chain · Mixing time

Mathematics Subject Classification 65C60 · 68R05 · 05A05

1 Introduction

Derangements are permutations $\sigma = \sigma_1 \cdots \sigma_n$ on integer $n \geq 2$ labels such that $\sigma_i \neq i$ for all $i = 1, \ldots, n$. Derangements are useful in a number of applications like in the testing of software branch instructions and random paths and data randomization and experimental design (Bacher et al. 2017; Diaconis et al. 2001; Sedgewick 1977). A well-known algorithm to generate random derangements is Sattolo's algorithm, that outputs a random cyclic derangement in $O(n)$ time (Gries and Xue 1988; Prodinger 2002; Sattolo 1986; Wilson 2009). An $O(2n)$

Communicated by Eduardo Souza de Cursi.

✉ J. Ricardo G. Mendonça
jricardo@usp.br

[1] Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, SP 03828-000, Brazil

algorithm to generate random derangements in general (not only cyclic derangements) has been given by Martínez et al. (2008) and Panholzer et al. (2004). Algorithms to generate all n-derangements in lexicographic or Gray order have also been developed (Akl 1980; Baril and Vajnovszki 2004; Korsh and LaFollette 2004).

In this paper, we propose two procedures to generate random derangements with the expected distribution of cycle lengths: one based on the randomization of derangements by random restricted transpositions (a random walk in the set of derangements) and the other based on a simple sequential importance sampling scheme. The generation of restricted permutations by means of sequential importance sampling is closely related with the problem of estimating the permanent of a 0–1 matrix, an important problem in, e.g., graph theory, statistical mechanics, and experimental design (Beichl and Sullivan 1999; Brualdi and Ryser 1991; Diaconis et al. 2001). Simulations show that the randomization algorithm samples a derangement in $O(n^a \log n^2)$ time, where n is the size of the derangement and $a \simeq \frac{1}{2}$, while the sequential importance sampling algorithm does it in $O(n)$ time but with a small probability $O(1/n)$ of failing. The algorithms are straightforward to understand and implement and can be modified to perform related computations of interest in many areas.

Throughout the paper, we employ the expected distribution of cycle lengths to analyse the algorithms, because they are such fundamental invariants of permutations from which many other statistics can be derived, for instance, the expected number of ascents, descents, or transpositions, because they offer a sufficiently "aggregate," not too detailed quantity, and also because we have simple exact expressions for the probability of observing derangements with a given number of cycles with which we can compare the numerical data.

2 Mathematical preliminaries

Let us briefly recapitulate some notation and terminology on permutations. Detailed accounts suited to our needs are given by Arratia et al. (2003) and Charalambides (2002).

We denote a permutation of a set of integer $n \geq 2$ labels (an n-permutation), formally a bijection of $[\,n\,] = \{1, \ldots, n\}$ onto itself, by $\sigma = \sigma_1 \cdots \sigma_n$, where $\sigma_i = \sigma(i)$. If σ and π are two n-permutations, their product is given by the composition $\sigma\pi = \sigma(\pi_1) \cdots \sigma(\pi_n)$. A cycle of length $k \leq n$ in a n-permutation σ is a sequence of indices $i_1, \ldots, i_k$ such that $\sigma(i_1) = i_2, \ldots, \sigma(i_{k-1}) = i_k$, and $\sigma(i_k) = i_1$, completing the cycle. Fixed points are 1-cycles, transpositions are 2-cycles. An n-permutation with a_k cycles of length k, $1 \leq k \leq n$, is said to be of type $(a_1, \ldots, a_n)$, with $\sum_k ka_k = n$. For example, the 9-permutation $174326985 = (1)(43)(6)(8)(9527)$ has 5 cycles and is of type $(3, 1, 0, 1)$, where we have omitted the trailing $a_5 = \cdots = a_9 = 0$. Note that in the cycle notation $(1)(43)(6)(8)(9527)$, the parentheses are unnecessary, since each new left-to-right absolute maximum $\sigma_i > \max\{\sigma_1, \ldots, \sigma_{i-1}\}$ corresponds to a new cycle (the so-called Foata's representation).

The number of n-permutations with k cycles is given by the unsigned Stirling number of the first kind $\left[{n \atop k}\right]$. Useful formulae involving these numbers are $\left[{0 \atop 0}\right] = 1$, $\left[{n \atop 0}\right] = 0$, and the recursion relation $\left[{n+1 \atop k}\right] = n\left[{n \atop k}\right] + \left[{n \atop k-1}\right]$. We have $\left[{n \atop n}\right] = 1$, counting just the identity permutation $\mathrm{id} = (1)(2)\cdots(n)$, $\left[{n \atop n-1}\right] = \binom{n}{2}$, counting n-permutations with $n-2$ fixed points, that can be taken in $\binom{n}{n-2} = \binom{n}{2}$ different ways, plus a transposition of the remaining two labels, and $\left[{n \atop 1}\right] = (n-1)!$, the number of cyclic n-derangements. It can also be shown that $\left[{n \atop 2}\right] = (n-1)!H_{n-1}$, where $H_k = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$ is the kth harmonic number. Obviously, $\left[{n \atop 1}\right] + \cdots + \left[{n \atop n}\right] = \left[{n+1 \atop 1}\right] = n!$, the total number of n-permutations.

Let us denote the set of all n-derangements by D_n. It is well known that

$$d_n = |D_n| = n!\Big(1 - \frac{1}{1!} + \cdots + \frac{(-1)^n}{n!}\Big) = \Big\lfloor \frac{n!+1}{e} \Big\rfloor, \quad n \geq 1, \tag{1}$$

the *rencontres* numbers, where the floor function $\lfloor x \rfloor$ evaluates to the greatest integer less than or equal to x. Let us also denote the set of k-cycle n-derangements, irrespective of their type, by $D_n^{(k)}$. The $D_n^{(k)}$ are disjoint with $D_n^{(k)} = \varnothing$ for $k > \lfloor n/2 \rfloor$. If we want to generate random n-derangements over $D_n = D_n^{(1)} \cup \cdots \cup D_n^{(\lfloor n/2 \rfloor)}$, we must be able to generate k-cycle random n-derangements with probabilities

$$\mathbb{P}(\sigma \in D_n^{(k)}) = \frac{d_n^{(k)}}{d_n}, \tag{2}$$

where $d_n^{(k)} = |D_n^{(k)}|$. The following proposition establishes the cardinality of the sets $D_n^{(k)}$.

Proposition 1 *The cardinality of the set $D_n^{(k)}$ is given by*

$$d_n^{(k)} = \sum_{j=0}^{k} (-1)^j \binom{n}{j} \left[{n-j \atop k-j}\right]. \tag{3}$$

Proof The number of n-permutations with k cycles is $\left[{n \atop k}\right]$. Of these, $n\left[{n-1 \atop k-1}\right]$ have at least one fixed point, $\binom{n}{2}\left[{n-2 \atop k-2}\right]$ have at least two fixed points, and so on. Perusal of the inclusion–exclusion principle furnishes the result. □

Proposition 2 *The numbers $d_n^{(k)}$ obey the recursion relation*

$$d_{n+1}^{(k)} = n\big(d_n^{(k)} + d_{n-1}^{(k-1)}\big) \tag{4}$$

with $d_0^{(0)} = 1$ and $d_n^{(0)} = 0$, $n \geq 1$.

Proof We give a combinatorial argument. Consider the derangement of $n+1$ labels with k cycles enumerated by $d_{n+1}^{(k)}$ according to the condition of the largest label $n+1$. This largest label is either in a 2-cycle or it is not. If it is, it is attaching a 2-cycle to an $n-1$-derangement with $k-1$ cycles, of which there are $d_{n-1}^{(k-1)}$, and since it can be paired up with any of other n possible labels, it contributes a factor $n\, d_{n-1}^{(k-1)}$ to $d_{n+1}^{(k)}$. If label $n+1$ is not in a 2-cycle, then it belongs to one of the k cycles of an otherwise n-derangement, of which there are $d_n^{(k)}$, and since in each cycle of length r, there are r possible places to insert label $n+1$ (the first and the last places within each cycle coincide) and $\sum r = n$, it contributes a factor $n\, d_n^{(k)}$ to $d_{n+1}^{(k)}$. Adding the two contributions furnishes the recursion relation (4). □

The numbers $d_n^{(k)}$ are sometimes called associated Stirling numbers of the first kind. Equation (3) recovers $d_n^{(0)} = 0$ and $d_n^{(1)} = \left[{n \atop 1}\right] = (n-1)!$ for $n \geq 1$, while we find that $d_n^{(2)} = (n-1)!(H_{n-2} - 1)$ for $n \geq 2$. Equation (4) generalizes the recursion relation $d_{n+1} = n(d_n + d_{n-1})$ for the *rencontres* numbers. A notable identity, valid for n even, is $d_n^{(n/2)} = (n-1)(n-3)\cdots 3 \cdot 1$, the number of fixed-point-free involutions σ such that $\sigma^2 = \mathrm{id}$, a. k. a. perfect matchings, see Sect. 3.2. From Eqs. (1)–(3), we see that already for small n we obtain $\mathbb{P}(\sigma \in D_n^{(1)}) \simeq e/n$ and $\mathbb{P}(\sigma \in D_n^{(2)}) \simeq (H_{n-2} - 1)e/n$.

Remark 1 One could consider the distribution of n-derangements over possible cycle types (instead of cycle lengths) for a "finer" view of the distribution. The number of n-permutations of type $(a_1, \dots, a_n)$ is given by Cauchy's formula

$$k_n(a_1, \dots, a_n) = \frac{n!}{1^{a_1} a_1! \cdots n^{a_n} a_n!}. \tag{5}$$

The analogue of (2) is given by $\mathbb{P}(\sigma \in K_n(0, a_2, \dots, a_n)) = k_n(0, a_2, \dots, a_n)/d_n$, where $K_n(0, a_2, \dots, a_n)$ is the conjugacy class formed by all n-permutations of type $(0, a_2, \dots, a_n)$. For example, for cyclic derangements $a_n = 1$ and all other $a_k = 0$, and we obtain $\mathbb{P}(\sigma \in K_n(0, \dots, 1)) = k_n(0, \dots, 1)/d_n \simeq e/n$, as expected.

Other permutation statistics, e.g. the expected number of descents or transpositions (every cycle of length k factors into $k-1$ transpositions, though) could be employed as well (see, for instance, (Diaconis and Holmes 2002, Sec. 1.5) for an interesting connection with integer partitions) but they would lead to more complicate expressions for $\mathbb{P}(\sigma \in \cdot\,)$.

3 Generating random derangements by random transpositions

3.1 The random transposition walk

Our first approach to generate random n-derangements consists in taking an initial n-derangement and to scramble it by random restricted transpositions enough to obtain a sample distributed over D_n according to the probabilities given by (2). By restricted transpositions, we mean swaps $\sigma_i \leftrightarrow \sigma_j$ avoiding pairs for which $\sigma_i = j$ or $\sigma_j = i$. Algorithm T describes the generation of random n-derangements according to this idea, where *mix* is a constant establishing the amount of random restricted transpositions to be attempted and *rnd* is a computer generated pseudorandom uniform deviate in $(0, 1)$.

Remark 2 Algorithm T is applicable only for $n \geq 4$, since it is not possible to connect the even permutations 231 and 312 by a single transposition.

A good choice for the initial derangement in Algorithm T is any cyclic derangement (cycle length $k = 1$), for example, $\sigma = (2\,3 \cdots n\,1)$. A particularly bad choice would be an involution (n even, all cycle lengths $k = 2$), for example, $\sigma = (n\; n-1) \cdots (2\;1)$, because then the algorithm would not be able to generate derangements with $k \neq 2$. Incidentally, this suggests the use of Algorithm T to generate random fixed-point-free involutions, a. k. a. random perfect matchings, see Sect. 3.2. To avoid this problem, we hardcoded the requirement that Algorithm T starts with a cyclic derangement. If several parallel streams of random derangements are sought, one can set different initial random cyclic derangements from a one-line implementation of Sattolo's algorithm.

Remark 3 The minimum number of restricted transpositions necessary to take a cyclic n-derangement into a k-cycle n-derangement is $k-1$, $1 \leq k \leq \lfloor n/2 \rfloor$, since transpositions of labels that belong to the same cycle split it into two cycles,

$$(ab)(i_1 \cdots i_{a-1} i_a i_{a+1} \cdots i_{b-1} i_b i_{b+1} \cdots i_k) = (i_1 \cdots i_{a-1} i_b i_{b+1} \cdots i_k)(i_{a+1} \cdots i_{b-1} i_a) \tag{6}$$

and, conversely, transpositions involving labels of different cycles join them into a single cycle. If Algorithm T is started with a cyclic derangement then one must set $mix \geq n/2$.

Algorithm T ensures that for a sufficiently large constant *mix* it generates a "sufficiently random" derangement from D_n. Slurring over the philosophical questions related with the

Algorithm T Random derangements by random restricted transpositions

```
Require: Initial cyclic $n$-derangement $\sigma_1\sigma_2\cdots\sigma_n$
$mix \leftarrow$ number of restricted transpositions to attempt
for $m = 1$ to $mix$ do
    $i \leftarrow \lceil rnd \cdot n \rceil$, $j \leftarrow \lceil rnd \cdot n \rceil$
    if $(\sigma_i \neq j) \wedge (\sigma_j \neq i)$ then
        swap $\sigma_i \leftrightarrow \sigma_j$
    end if
end for
Ensure: For sufficiently large $mix$, $\sigma_1 \cdots \sigma_n$ is a "sufficiently random" derangement from $D_n$
```

concept of randomness, in this paper "sufficiently random" means with cycle lengths distributed according to the exact probabilities given by Eq. (2). We make this statement about Algorithm T more precise in Sect. 5. Clearly, the correct distribution of cycle lengths is a necessary but not sufficient condition for uniformity ($\mathbb{P}(\sigma \in D_n) = 1/d_n$) over D_n, and we do not claim uniformity for Algorithm T here or elsewhere in this paper. This point is elaborated further in Remark 4 and Sect. 4.3.

We run Algorithm T for $n = 64$ and different values of $mix \geq n$ and collect data. Simulations were performed on Intel Xeon E5-1650 v3 processors running `-O3` compiler-optimized C code (GCC v. 7.4.0) over Linux kernel 4.15.14 at 3.50 GHz, while the numbers (3) were calculated on the software package Mathematica 11.3 (Wolfram 2018). We draw our pseudorandom numbers from Vigna's superb `xoshiro256+` generator (Vigna 2019). Our results appear in Table 1. We see from that table that with $mix = n$ random restricted transpositions there is a slight excess of probability mass in the lower k-cycle sets with $k = 1, 2$, and 3. Trying to scramble the initial n-derangement by $2n$ restricted transpositions performs better. The difference between attempting $2n$ and $n \log n$ random restricted transpositions is much less pronounced. Figures for derangements of higher cycle number fluctuate more due to the finite size of the sample. The data suggest that Algorithm T can generate a random n-derangement uniformly distributed on D_n with $2n$ random restricted transpositions, employing $4n$ pseudorandom numbers in the process. This is further discussed in Sect. 5.

Remark 4 It is a classic result that $O(n \log n)$ transpositions are needed before an unrestricted shuffle by transpositions becomes "sufficiently random" (Aldous and Diaconis 1986; Diaconis and Shahshahani 1981). A similar analysis for random transpositions over derangements is complicated by the fact that derangements do not form a group. Recently, the analysis of the spectral gap of the Markov transition kernel of the process provided the upper bound $mix < Cn + an \log n^2$, with $a > 0$ and $C \geq 0$ a decreasing function of n (Smith 2015). This bound results from involved estimations and approximations and may not be very accurate. Related results for the mixing time of the random transposition walk over permutations with one-sided restrictions $\sigma_i \geq b_i$ for given $n \geq b_n \geq \cdots \geq b_1 \geq 1$—a pattern known as a Ferrer's board in the combinatorics literature—appear in (Blumberg 2012; Hanlon 1996). Recently, the case $i - t \leq \sigma_i \leq i + 1$, $t \geq 1$, has been treated by Chung et al. (2019), although they do not explore the mixing times of the associated Markov chains.

3.2 The perfect matching connection

In Sect. 3.1, we remarked that if one seeds Algorithm T with an initial fixed-point-free involution, i.e., a derangement with all cycle lengths equal to 2, then all subsequent derangements generated by the algorithm will also be fixed-point-free involutions. Such derangements are

Table 1 Proportion of n-derangements in $D_n^{(k)}$ measured in 10^{10} samples generated by Algorithms T and S for $n = 64$

Cycles	Algorithm T (*mix*)			Algorithm S	Exact
k	n	$2n$	$n \log n$	–	Eqs. (1)–(3)
1	0.042 933	0.042 479	0.042 473	0.042 475	0.042 473
2	0.158 395	0.157 691	0.157 679	0.157 684	0.157 677
3	0.260 129	0.258 787	0.258 765	0.258 788	0.258 772
4	0.252 739	0.253 304	0.253 305	0.253 306	0.253 301
5	0.167 189	0.167 621	0.167 639	0.167 622	0.167 635
6	0.079 498	0.080 390	0.080 402	0.080 389	0.080 400
7	0.028 825	0.029 192	0.029 195	0.029 196	0.029 200
8	0.008 087	0.008 269	0.008 274	0.008 272	0.008 274
9	0.001 821	0.001 868	0.001 869	0.001 868	0.001 869
10	3.292_{-4}	3.416_{-4}	3.418_{-4}	3.412_{-4}	3.417_{-4}
11	4.914_{-5}	5.109_{-5}	5.120_{-5}	5.103_{-5}	5.116_{-5}
12	5.997_{-6}	6.322_{-6}	6.301_{-6}	6.354_{-6}	6.326_{-6}
13	6.215_{-7}	6.493_{-7}	6.301_{-7}	6.507_{-7}	6.499_{-7}
14	4.83_{-8}	5.40_{-8}	5.57_{-8}	5.44_{-8}	5.569_{-8}
15	4.6_{-9}	3.1_{-9}	3.0_{-9}	4.1_{-9}	3.989_{-9}
16	4_{-10}	1_{-10}	3_{-10}	1_{-10}	2.390_{-10}

The notation x_{-a} reads $x \times 10^{-a}$. Data for Algorithm S are based on a run with a ratio of completed/attempted derangements of 0.985472

in 1–1 correspondence with perfect matchings on a complete graph, since any unoriented edge $\sigma_i \sigma_j$ can occur. A perfect matching on a graph is a set of disjoint edges of the graph containing all its vertices. The connection between permutations with restricted positions and perfect matchings is well known (Brualdi and Ryser 1991; Lovász and Plummer 2009) and has been explored recently in the context of random walks on trees and applications, including Monte Carlo estimation of hard enumeration problems (Chung et al. 2019; Diaconis and Holmes 1998, 2002; Diaconis and Kolesnik 2019; Dyer et al. 2017; Dyer and Müller 2019).

Cauchy's formula (5) gives the number of perfect matchings on a complete graph of even number n of vertices as the number of derangements with $n/2$ cycles of length 2,

$$k_n(0, n/2, 0, \ldots, 0) = \frac{n!}{2^{n/2}(n/2)!} \sim \sqrt{2(n/e)^n}, \tag{7}$$

where the asymptotics follows from Stirling's approximation $n! \simeq \sqrt{2\pi n}\,(n/e)^n$. The number (7) can also be understood as the number of partitions of a set of even size n into $n/2$ unordered parts of size 2 each—which is just another definition of a perfect matching. We see that the probability that a random derangement is a perfect matching is very small,

$$\mathbb{P}(\sigma \in D_n^{(n/2)}) = \frac{k(0, n/2, 0, \ldots, 0)}{d_n} \simeq \frac{e}{\sqrt{\pi n}} \sqrt{(e/n)^n}. \tag{8}$$

For example, for $n = 10$ Eq. (8) gives a 1 in 1389 chance that a random derangement is a perfect matching. If one employs a standard algorithm to generate random permutations, the chance that it outputs a random perfect matching decreases to 1 in 3777. With a simple tweak, though, Algorithm T can generate random perfect matchings on the complete graph

Algorithm S Random derangements by sequential importance sampling

```
1: $J \leftarrow [\,n\,]$
2: for $i = 1$ to $n$ do
3:     $J_i \leftarrow J \setminus \{i\}$
4:     if $J_i \neq \varnothing$ then
5:         choose $j_i \in J_i$ uniformly at random
6:         $\sigma_i \leftarrow j_i$
7:         $J \leftarrow J \setminus \{j_i\}$
8:     else
9:         fail
10:    end if
11: end for
Ensure: If completed, $\sigma_1 \cdots \sigma_n$ is a “sufficiently random” derangement from $D_n$
```

at will. Although this is not a particularly difficult computational problem, having a simple and efficient algorithm to generate such random perfect matchings might be useful.

4 Sequential importance sampling of derangements

4.1 The SIS algorithm

Sequential importance sampling (SIS) is an importance sampling scheme with the sampling weights built up sequentially. The idea is particularly suited to sample composite objects $X = X_1 \cdots X_n$ from a complicated sample space $\mathscr{X}$ for which the high-dimensional volume $|\mathscr{X}|$, from which the uniform distribution $\mathbb{P}(X) = |\mathscr{X}|^{-1}$ follows, may not be easily calculable. However, since we can always write

$$\mathbb{P}(X_1 \cdots X_n) = \mathbb{P}(X_1)\mathbb{P}(X_2 \mid X_1) \cdots \mathbb{P}(X_n \mid X_1 \cdots X_{n-1}), \tag{9}$$

we can think of “telescoping” the sampling of X by first sampling X_1, then use the updated information brought by the knowledge of X_1 to sample X_2 and so on. In Monte Carlo simulations, the right-hand side of (9) actually becomes $\mathbb{P}_1(X_1)\mathbb{P}_2(X_2 \mid X_1) \cdots \mathbb{P}_n(X_n \mid X_1 \cdots X_{n-1})$, with the distributions $\mathbb{P}_i(\,\cdot\,)$ estimated or inferred incrementally based on approximate weighting functions for the partial objects $X_1 \cdots X_{i-1}$. Expositions of the SIS framework of interest to what follows appear in Chen et al. (2005) and Diaconis et al. (2001).

Algorithm S describes a SIS algorithm to generate random derangements inspired by the analogous problem of sampling contingency tables with restrictions (Chen et al. 2005; Diaconis et al. 2001) as well as by the problem of estimating the permanent of a matrix (Beichl and Sullivan 1999; Chen et al. 2005; Kuznetsov 1996; Rasmussen 1994). Our presentation of Algorithm S is not the most efficient for implementation; the auxiliary sets J_i, for instance, are not actually needed and were included only to facilitate the analysis of the algorithm, and the n tests in line 4 can be reduced to a single test in the last pass, since all $J_i \neq \varnothing$ except perhaps J_n.

The distribution of cycle lengths in 10^{10} derangements generated by Algorithm S is presented in Table 1. We see excellent agreement between the data and the expected values.

4.2 Failure probability of the SIS algorithm

In the ith pass of the loop in Algorithm S, σ_i can pick (lines 5 and 6) one of either $n-i$ or $n-i+1$ labels, depending on whether label i has already been picked. This guarantees the construction of the n-derangement up to the $(n-1)$st label σ_{n-1}. The n-derangement is completed only if the last remaining label is different from n, such that σ_n does not pick n. The probability that Algorithm S fails is thus given by

$$\mathbb{P}(\sigma_n = n \mid \sigma_1 \cdots \sigma_{n-1}) = \mathbb{P}(\sigma_1 \neq n)\, \mathbb{P}(\sigma_2 \neq n \mid \sigma_1) \cdots \mathbb{P}(\sigma_{n-1} \neq n \mid \sigma_1 \cdots \sigma_{n-2}). \quad (10)$$

Now, according to Algorithm S, line 5, we have

$$\mathbb{P}(\sigma_i \neq n \mid \sigma_1 \cdots \sigma_{i-1}) = 1 - \mathbb{P}(\sigma_i = n \mid \sigma_1 \cdots \sigma_{i-1}) = 1 - \frac{1}{\mathbb{E}(|J_i(\sigma_1 \cdots \sigma_{i-1})|)}, \quad (11)$$

where $\mathbb{E}(|J_i(\sigma_1 \cdots \sigma_{i-1})|)$ is the expected size of the set J_i in the ith pass of the loop in Algorithm S. The failure probability then becomes

$$\mathbb{P}(\sigma_n = n \mid \sigma_1 \cdots \sigma_{n-1}) = \prod_{i=1}^{n-1} \Big(1 - \frac{1}{E_i}\Big), \quad (12)$$

where E_i stands for $\mathbb{E}(|J_i(\sigma_1 \cdots \sigma_{i-1})|)$. Expression (12) is but a probabilistic version of the inclusion–exclusion principle.

The computation of (12) is a cumbersome business and will not be pursued here. The following theorem establishes an easy upper bound on the failure probability of Algorithm S.

Theorem 1 *Algorithm S fails with probability $O(1/n)$.*

Proof In the ith pass of the loop in Algorithm S we have

$$|J_i(\sigma_1 \cdots \sigma_{i-1})| = n - i + \sum_{j=1}^{i-1} \mathbb{1}\{(\sigma_j = i \mid \sigma_1, \ldots, \sigma_{j-1})\}, \quad (13)$$

where the symbol $\mathbb{1}\{A\}$ equals 1 if A occurs and 0 if A does not occur. We thus have that $|J_i(\sigma_1 \cdots \sigma_{i-1})| = n - i$ or $n - i + 1$, such that the expectation $E_i = \mathbb{E}(|J_i(\sigma_1 \cdots \sigma_{i-1})|)$ obeys

$$1 - \frac{1}{n-i} < 1 - \frac{1}{E_i} < 1 - \frac{1}{n-i+1} \quad (14)$$

and it immediately follows that

$$\mathbb{P}(\sigma_n = n \mid \sigma_1 \cdots \sigma_{n-1}) = \prod_{i=1}^{n-1} \Big(1 - \frac{1}{E_i}\Big) < \prod_{i=1}^{n-1} \Big(1 - \frac{1}{n-i+1}\Big) = \frac{1}{n}. \quad (15)$$

□

We can obtain a slightly better bound for $\mathbb{P}(\sigma_n = n \mid \sigma_1 \cdots \sigma_{n-1})$ as follows. The difficulty in the calculation of $\mathbb{E}(|J_i(\sigma_1 \cdots \sigma_{i-1})|)$ resides in the calculation of $\mathbb{E}(\mathbb{1}\{(\sigma_j = i \mid \sigma_1, \ldots, \sigma_{j-1})\})$. We can approximate this calculation by ignoring the conditioning of the event $(\sigma_j = i)$ on the event $(\sigma_1 \cdots \sigma_{j-1})$, i.e., by ignoring correlations between the values assumed by the σ_j along a "path" $\sigma_1 \to \sigma_2 \to \cdots$ in the algorithm. The approximation is

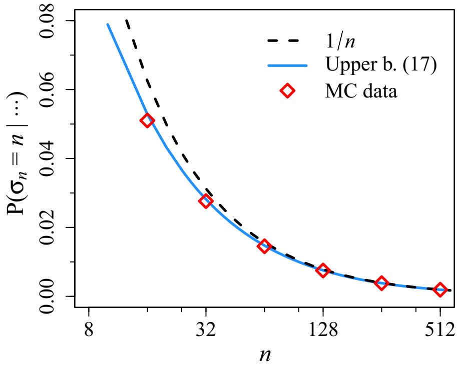

Fig. 1 Measured failure rate for Algorithm S against $1/n$ and the upper bound (17). Error bars in the data are much smaller than the symbols shown

clearly better in the beginning of the construction of σ, when j is small and there is not yet much conditioning going on, than later. We get

$$\begin{aligned}\mathbb{E}(|J_i(\sigma_1\cdots\sigma_{i-1})|) &= n-i+\sum_{j=1}^{i-1}\mathbb{E}\big(\mathbb{1}\{(\sigma_j = i \mid \sigma_1,\ldots,\sigma_{j-1})\}\big)\\ &\approx n-i+\sum_{j=1}^{i-1}\mathbb{E}\big(\mathbb{1}\{\sigma_j = i\}\big) = n-i+\frac{i-1}{n-1}.\end{aligned} \tag{16}$$

This approximate E_i is greater than the true E_i, because conditioning J_i on $(\sigma_1\cdots\sigma_{i-1})$ can only restrict the set of indices available to σ_i, not enlarge it. The approximate value of $1-1/E_i$ is thus greater than its true value, and we can bound the failure probability (12) by

$$\mathbb{P}(\sigma_n = n \mid \sigma_1\cdots\sigma_{n-1}) < \prod_{i=1}^{n-1}\left(1-\frac{1}{n-i+\frac{i-1}{n-1}}\right) = \frac{1}{n-1}\prod_{i=1}^{n-1}\left[1+\frac{1}{(n-2)(n-i)}\right]^{-1}. \tag{17}$$

The measured failure rate for the SIS data in Table 1 is $1-0.985472 = 0.014528$, not far from $1/64 = 0.015625$. A sample of 10^4 runs of Algorithm S of 10^6 derangements each with $n = 64$ gives an average failure rate of 0.01453(12) with a sample minimum of 0.014130 and maximum of 0.014991, where the digits within parentheses indicate the uncertainty at one standard deviation in the corresponding last digits of the datum. Figure 1 depicts Monte Carlo data for the failure probability (12) against the upper bounds $1/n$ and (17). Each data point was obtained as an average over 10^4 runs of Algorithm S of 10^6 derangements each except for $n = 512$, for which the runs are of 2×10^5 derangements each.

4.3 Uniformity (or the lack thereof) of the SIS algorithm

In the SIS approach, the ensuing sampling probabilities may deviate considerably from the uniform distribution. As it happens, Algorithm S does not generate each derangement in D_n with uniform probability $1/d_n$. This can be seen by a simple pencil-simulation of the algorithm for some small n, say $n = 3$. In this case $D_3 = \{231, 312\}$. To build 312, we must first pick 3 with probability 1/2 then choose 1 and 2 sequentially, thus generating the derangement 312 with probability 1/2. To build 231, otherwise, we must first pick 2 with probability 1/2, then 3 with probability 1/2 and then 1 is forced, such that 231 occurs with

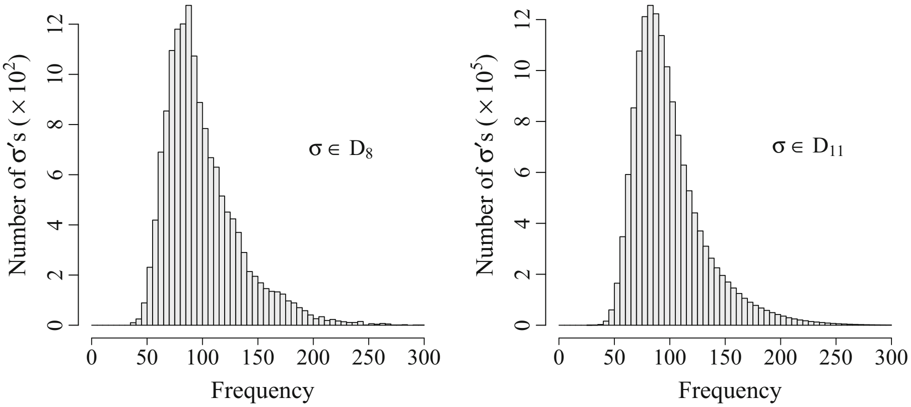

Fig. 2 Number of derangements $\sigma \in D_n$ that occur in a sample of size $100d_n$ generated by Algorithm S for $n = 8$ and 11. The bins are of size 5. Note the different vertical scales, as $d_{11}/d_8 \simeq 990$. If Algorithm S sampled D_n uniformly, we would expect a sharp peak of height $O(d_n)$ located at frequency 100

probability 1/4. If we first pick 2 and then 1 the algorithm fails with probability 1/4. We see that, by the rules of Algorithm S, $\mathbb{P}(\sigma = 312) \neq \mathbb{P}(\sigma = 231)$.

To verify whether the probability imbalance persists or smoothes out for larger n, we generate $100\,d_n$ derangements by Algorithm S for $n = 8$ ($d_8 = 14833$) and $n = 11$ ($d_{11} = 14\,684\,570$) and bin the data. It is hard to run statistical tests involving all derangements for $n > 11$ because either the sizes of the data files become humongous (hundreds of gigabytes if we insist in $100\,d_n$ samples) or the processing time becomes prohibitive (e.g., binning the derangements on the run involves searching). We found that Algorithm S indeed generated all derangements in D_8 and D_{11} many times each in the runs. Figure 2, however, definitely does not depict a distribution of occurrences peaked sharply about 100 (the bins are of size 5) which would represent uniform distribution. We cannot even argue that the distributions are becoming sharper with increasing n, since the standard deviation of the data are virtually the same in both cases: $\mathrm{sd}_8 \simeq 33.5$ versus $\mathrm{sd}_{11} \simeq 32.4$.

Yet the data in Table 1 clearly suggest that Algorithm S does sample D_n according to the expected distribution of cycle lengths for $n = 64$; the same behavior was also observed for a couple of other $n \geq 20$. A possible explanation is that D_n is so large already for moderate values of n (for instance, $d_{20} = 8.950 \times 10^{17}$), that any relatively "small" sample (which can actually be extremely large in absolute numbers) obtained by Algorithm S will most likely not contain repeated derangements. We verified this claim empirically: in five separate samples of 10^8 derangements of 20 labels each, not a single derangement occurred twice either within a sample or between them. For practical purposes, then, Algorithm S samples D_n "uniformly."

We could neither prove the uniformity nor the non-uniformity of Algorithm S rigorously. An attempt based on techniques borrowed from Beichl and Sullivan (1999); Chen et al. (2005); Kuznetsov (1996); Rasmussen (1994) proved flawed. For one-sided restricted permutations of the type $\sigma_i \geq b_i$ for given $n \geq b_n \geq \cdots \geq b_1 \geq 1$ (cf. Remark 4), Diaconis et al. (2001) prove (Lemma 3.2) that a simple SIS algorithm samples all possible permutations uniformly and, moreover, that the algorithm never fails because of the particular form of the restrictions. A recent account on the SIS approach to sample one-sided restricted permutations is given by Chung et al. (2019). Their arguments do not seem to apply to derangements, though.

5 Mixing time of the restricted transpositions shuffle

To shed some light on the question of how many random restricted transpositions are necessary to generate random derangements uniformly over D_n, we investigate the convergence of Algorithm T numerically. This can be done by monitoring the evolution of the empirical probabilities along the run of the algorithm towards the exact probabilities given by (2).

Let ν be the measure that puts mass $\nu(k) = d_n^{(k)}/d_n$ on the set $D_n^{(k)}$ and μ_t be the empirical measure

$$\mu_t(k) = \frac{1}{t}\sum_{s=1}^{t} \mathbb{1}\{\sigma_s \in D_n^{(k)}\}, \tag{18}$$

where σ_s is the derangement obtained after attempting s restricted transpositions by Algorithm T on a given initial derangement σ_0. The total variance distance between μ_t and ν is given by (Aldous and Diaconis 1986; Diaconis 1988)

$$d_{\mathrm{TV}}(t) = \|\mu_t - \nu\|_{\mathrm{TV}} = \frac{1}{2}\sum_{k=1}^{\lfloor n/2 \rfloor} |\mu_t(k) - \nu(k)|. \tag{19}$$

The right-hand side of (19) can be seen as the "histogram distance" between μ_t and ν in the ℓ_1 norm. Clearly, $0 \leq d_{\mathrm{TV}}(t) \leq 1$. This distance allows us to define $t_{\mathrm{mix}}(\varepsilon)$ as the time it takes for μ_t to fall within distance ε of ν,

$$t_{\mathrm{mix}}(\varepsilon) = \min\{t \geq 0 \colon d_{\mathrm{TV}}(t) < \varepsilon\}. \tag{20}$$

It is usual to define *the* mixing time t_{mix} by setting $\varepsilon = \frac{1}{4}$ or $\varepsilon = \frac{1}{2}e^{-1} \simeq 0.184$, this last figure being reminiscent of the spectral analysis of Markov chains. We set $\varepsilon = \frac{1}{2}e^{-1}$. This choice is motivated by the following pragmatic reasons:

(i) We want the derangements output by Algorithm T to be as uniformly distributed over D_n as possible, so the smaller the ε the better the assessment of the algorithm and the choice of the constant *mix*;
(ii) Most of the probability mass is concentrated on a few cycle numbers (see Table 1 and Remark 5 below), such that even relatively small differences between μ_t and ν are likely to induce noticeable biases in the output of Algorithm T;
(iii) With $\varepsilon = \frac{1}{4}$ we found that $t_{\mathrm{mix}} < n/2$, meaning that not even every possible derangement had chance to be generated if the initial derangement is cyclic (see Remark 3).

Remark 5 It is well known that the number of k-cycles of random n-permutations is Poisson distributed with mean $1/k$, such that as $n \nearrow \infty$ the CLT implies that the length of the cycles of random permutations follow a normal distribution with mean $\log n$ and variance $\log n$; see, e.g., Arratia et al. (2003) and the references there in. Flajolet and Soria (1990) proved that the same holds for permutations with no cycles of length less than a given $\ell > 1$ using complex asymptotics of exponential generating functions; Martínez et al. (2008) and Panholzer et al. (2004) provide the analysis for the particular case of derangements. Figure 3 displays the exact distribution of k-cycles for derangements with $n = 2^{15} = 32768$ together with the normal density $N(\log n, \sqrt{\log n})$. For $n = 32768$, we obtain from Eqs. (2)–(3) that $\langle k \rangle = 9.967\cdots$ and $\sqrt{\langle k^2 \rangle - \langle k \rangle^2} = 2.872\cdots$, while $\log n = 10.397\cdots$ and $\sqrt{\log n} = 3.224\cdots$. The distribution of cycle lengths in Fig. 3 indeed looks close to a normal $N(\log n, \sqrt{\log n})$, albeit slightly skewed. We did not go to greater n because Stirling numbers of the first kind are notoriously hard to compute even by computer algebra systems running on modern

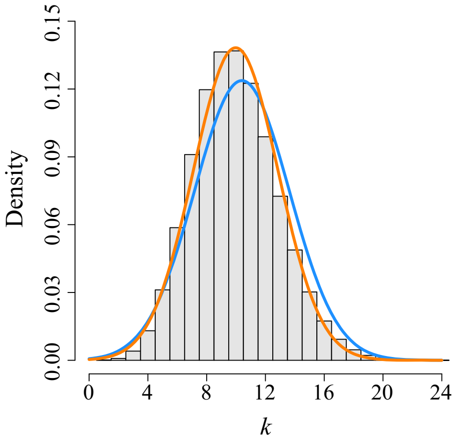

Fig. 3 Distribution of cycle lengths of random n-derangements for $n = 2^{15} = 32768$ together with the normal densities $N(\log n, \sqrt{\log n})$ (shorter, in blue) and $N(m, s)$ (taller, in orange) with $m = \langle k \rangle = 9.967 \cdots$ and $s = \sqrt{\langle k^2 \rangle - \langle k \rangle^2} = 2.872 \cdots$ (Color figure online)

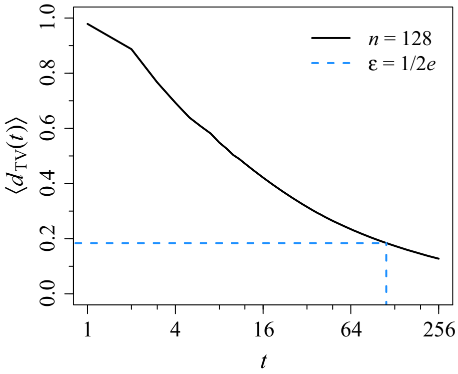

Fig. 4 Total variance distance $\langle d_{\mathrm{TV}}(t) \rangle$ (averaged over 10^6 runs) between the empirical measure μ_t (with $\mu_0(1) = 1$) and the stationary measure ν of the process defined by Algorithm T for $n = 128$. The dashed line indicates the level $\varepsilon = \frac{1}{2}e^{-1}$

workstations. Recently, the cycle structure of certain types of restricted permutations (with $\sigma_i \geq i - 1$) was also shown to be asymptotically normal (Ozel 2017).

Starting with a cyclic derangement, i.e., with $\mu_0(1) = 1$ and all other $\mu_0(k) = 0$, we run Algorithm T and collect statistics on $d_{\mathrm{TV}}(t)$. Figure 4 displays the average $\langle d_{\mathrm{TV}}(t) \rangle$ over 10^6 runs for $n = 128$. The behavior of $\langle d_{\mathrm{TV}}(t) \rangle$ does not show sign of the cutoff phenomenon—a sharp transition from unmixed state ($d_{\mathrm{TV}}(t_{\mathrm{mix}} - \delta) \approx 1$) to mixed state ($d_{\mathrm{TV}}(t_{\mathrm{mix}} + \delta) \approx 0$) over a small window of time $\delta \ll t_{\mathrm{mix}}$. Table 2 lists the average $\langle t_{\mathrm{mix}} \rangle$ obtained over 10^6 samples for larger derangements at $\varepsilon = \frac{1}{2}e^{-1}$. An adjustment of the data to the form

$$t_{\mathrm{mix}} = cn^a \log n^2 \tag{21}$$

furnishes $a = 0.527(2)$ and $c = 0.90(1)$. Our data thus suggest that $t_{\mathrm{mix}} \sim O(n^a \log n^2)$ with $a \simeq \frac{1}{2}$, roughly an $O(\sqrt{n})$ lower than the upper bound given by Smith (2015). It is tempting to conjecture that $a = \frac{1}{2}$ (and, perhaps, that $c = 1$) exactly, cf. last two lines of Table 2, although our data do not support the case unequivocally.

Table 2 Mixing time t_{mix} evaluated at $\varepsilon = \frac{1}{2}e^{-1}$ obtained from an average trajectory $\langle d_{\text{TV}}(t)\rangle$ over 10^6 samples; see Fig. 4

n	64	128	192	256	320	384	448	512
$\langle t_{\text{mix}}\rangle$	67	112	150	184	216	245	274	301
$\sqrt{n}\log n^2$	67	110	146	177	206	233	258	282
a in $n^a \log n^2$	0.502	0.504	0.505	0.507	0.508	0.508	0.510	0.510

The second line displays the best guess to $n^a \log n^b$ involving only integer and semi-integer exponents. The last line displays the adjusted a supposing a dependence like in (21) with $c = 1$

6 Summary and conclusions

While simple rejection-sampling generates random derangements with an acceptance rate of $\sim e^{-1} \simeq 0.368$, thus being $O(e{\cdot}n)$ (plus the cost of verifying if the permutation generated is a derangement, which does not impact the complexity of the algorithm but impacts its runtime), Sattolo's $O(n)$ algorithm only generates cyclic derangements, and Martínez–Panholzer–Prodinger algorithm, with guaranteed uniformity, is $2n + O(\log^2 n)$, we described two procedures, Algorithms T and S, that are competitive for the efficient generation of random derangements. In Sect. 3.2, we discussed how Algorithm T can also be used, with n even, to generate only random fixed-point-free involutions. Since fixed-point-free involutions of even n labels can be viewed as perfect matchings on the complete graph, Algorithm T can become handy in a multitude of situations.

We found, numerically, that $O(n^a \log n^2)$ random restricted transpositions with $a \simeq \frac{1}{2}$ suffice to spread an initial n-derangement over D_n measured by the distribution of cycle lengths. The fact that $2n > cn^a \log n^2$ for all $n \geq 1$ as long as $a \leq 0.63$ and $c \leq 1$ explains the good statistics displayed by Algorithm T with $mix = 2n$, see Table 1. Currently, there are few analytical results on the mixing time of the random restricted transposition walk implemented by Algorithm T; the upper bound $O(n \log n^2)$ obtained by Smith (2015) is roughly $O(\sqrt{n})$ above our numerical estimations. Diaconis and Holmes (1998, 2002) obtain a sharp $O(n \log n)$ estimate for the mixing time of a "switch Markov chain" for perfect matchings. Their chain builds perfect matchings as unordered sets $\{i, j\}$, not as ordered pairs (i, j), as we do. Their numbers, however, are clearly equal because as a 2-cycle $(ij) \equiv (ji)$. It would be interesting to run Algorithm T in the "perfect matchings mode" to check whether its mixing time display a different behavior.

Algorithm T employs $2\,mix$ pseudorandom numbers and Algorithm S employs $O(n)$ pseudorandom numbers to generate an n-derangement distributed over D_n with the expected distributions of cycle lengths. In this way, even if we set $mix = c\sqrt{n}\,\log n^2$ with some $1 < c \sim O(1)$, both algorithms perform better than currently known methods, with comparable runtime performances between them. As we argued in Sect. 4.3, for relatively small samples, which can actually be very large in absolute numbers (several billion derangements, for instance) since D_n is such a huge set already for moderate n, in practice Algorithm S samples derangements "uniformly."

Acknowledgements The author thanks Aaron Smith (U. Ottawa) for useful correspondence and suggestions improving a previous version of the manuscript, the Laboratoire de Physique Théorique et Modèles Statistiques – LPTMS (CNRS UMR 8486) for kind hospitality during a sabbatical leave in France where part of this work was done, and FAPESP (Brazil) for partial support through Grants nos. 2017/22166-9 and 2020/04475-7.

References

Akl SG (1989) A new algorithm for generating derangements. BIT Numer Math 20(1):2–7

Aldous D, Diaconis P (1986) Shuffling cards and stopping times. Am Math Mon 93(5):333–348

Arratia R, Barbour AD, Tavaré S (2003) Logarithmic combinatorial structures: a probabilistic approach. EMS, Zürich

Bacher A, Bodini O, Hwang H-K, Tsai T-H (2017) Generating random permutations by coin tossing: classical algorithms, new analysis, and modern implementation. ACM Trans Algorithms 13(2):24

Baril JL, Vajnovszki V (2004) Gray code for derangements. Discret Appl Math 140(1–3):207–221

Beichl I, Sullivan F (1999) Approximating the permanent via importance sampling with application to the dimer covering problem. J Comput Phys 149(1):128–147

Blumberg O (2012) Cutoff for the transposition walk on permutations with one-sided restrictions. arXiv:1202.4797 [math.PR]

Brualdi RA, Ryser RJ (1991) Combinatorial matrix theory. Cambridge University Press, Cambridge

Charalambides CA (2002) Enumerative combinatorics. Chapman & Hall/CRC, Boca Raton

Chen Y, Diaconis P, Holmes SP, Liu JS (2005) Sequential Monte Carlo methods for statistical analysis of tables. J Am Stat Assoc 100(469):109–120

Chung F, Diaconis P, Graham R (2019) Permanental generating functions and sequential importance sampling. Adv Appl Math. https://doi.org/10.1016/j.aam.2019.05.004 **(in press)**

Diaconis P (1988) Group representations in probability and statistics. IMS, Hayward

Diaconis PW, Holmes SP (1998) Matchings and phylogenetic trees. Proc Natl Acad Sci USA 95(25):14600–14602

Diaconis P, Holmes S (2002) Random walks on trees and matchings. Electron J Probab 7:6

Diaconis P, Kolesnik B (2019) Randomized sequential importance sampling for estimating the number of perfect matchings in bipartite graphs. arXiv:1907.02333 [math.PR]

Diaconis P, Shahshahani M (1981) Generating a random permutation by random transpositions. Z Wahrsch Verw Gebiete 57(2):159–179

Diaconis P, Graham RL, Holmes SP (2001) Statistical problems involving permutations with restricted positions. In: de Gunst M, Klaassen C, Van der Vaart A (eds) State of the art in probability and statistics: Festschrift for Willem R. van Zwet. IMS, Beachwood, pp 195–222

Dyer M, Müller H (2019) Counting perfect matchings and the switch chain. SIAM J Discret Math 33(3):1146–1174

Dyer M, Jerrum M, Müller H (2017) On the switch Markov chain for perfect matchings. J ACM 64(2):12

Flajolet P, Soria M (1990) Gaussian limiting distributions for the number of components in combinatorial structures. J Comb Theor Ser A 53(2):165–182

Gries D, Xue J (1988) Generating a random cyclic permutation. BIT Numer Math 28(3):569–572

Hanlon P (1996) A random walk on the rook placements on a Ferrer's board. Electron J Comb 3(2):26

Korsh JF, LaFollette PS (2004) Constant time generation of derangements. Inf Process Lett 90(4):181–186

Kuznetsov NY (1996) Computing the permanent by importance sampling method. Cybern Syst Anal 32(6):749–755

Lovász L, Plummer MD (2009) Matching theory. Corrected reprint. AMS, Providence

Martínez C, Panholzer A, Prodinger H (2008) Generating random derangements. In: Sedgewick R, Szpankowski W (eds) Proceedings of the fifth workshop on analytic algorithmics and combinatorics—ANALCO. SIAM, Philadelphia, pp 234–240

Ozel E (2017) The number of k-cycles in a family of restricted permutations. arXiv:1710.07885 [math.PR]

Panholzer A, Prodinger H, Riedel M (2004) Measuring post–quickselect disorder. J Iran Stat Soc 3(2):219–249

Prodinger H (2002) On the analysis of an algorithm to generate a random cyclic permutation. Ars Comb 65:75–78

Rasmussen LE (1994) Approximating the permanent: a simple approach. Random Struct Algorithms 5(2):349–361

Sattolo S (1986) An algorithm to generate a random cyclic permutation. Inf Process Lett 22(6):315–317

Sedgewick R (1977) Permutation generation methods. Comput Surv 9(2):137–164

Smith A (2015) Comparison theory for Markov chains on different state spaces and application to random walk on derangements. J Theor Probab 28(4):1406–1430

Vigna S (2019) xoshiro/xoroshiro generators and the PRNG shootout. http://xoshiro.di.unimi.it/. Accessed 15 Dec 2019

Wilson MC (2009) Random and exhaustive generation of permutations and cycles. Ann Comb 12(4):509–520
Wolfram Research, Inc. (2018) Mathematica, Version 11.3. Champaign

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.