RANDOM AND EXHAUSTIVE GENERATION OF PERMUTATIONS AND CYCLES

MARK C. WILSON

Abstract. In 1986 S. Sattolo introduced a simple algorithm for uniform random generation of cyclic permutations on a fixed number of symbols. This algorithm is very similar to the standard method for generating a random permutation, but is less well known.

We consider both methods in a unified way, and discuss their relation with exhaustive generation methods. We analyse several random variables associated with the algorithms and find their grand probability generating functions, which gives easy access to moments and limit laws.

1. The algorithms

Basic notation. For each $n \geq 1$, we denote by S_n the ***symmetric group*** (set of all permutations under the operation of composition) on the set $[n] := \{1, \ldots, n\}$. Suppose that $n \geq 2$. For each $\pi \in \mathrm{S}_{n-1}$, let π^* be its ***extension*** to S_n, by definition the element of S_n that fixes n and agrees with π on $[n-1]$. The map * is injective on each S_n. Dually, for an element $\rho \in \mathrm{S}_n$ that fixes n, ρ_* is the ***restriction*** of ρ to S_{n-1}. The map $_*$ is onto each S_{n-1} and * followed by $_*$ is the identity on S_{n-1}. Thus we may consider that $\mathrm{S}_n \subset \mathrm{S}_{n+1}$ and let S be the union of all S_n (formally, we consider the direct limit induced by the natural inclusion maps $[n] \to [n+1]$). Each element π of S belongs to a maximal S_n, where n is the largest integer moved by π; we define $n(\pi)$ to be this value of n.

The ***action*** of $\pi \in \mathrm{S}$ on i is denoted by $i\pi$. We use the standard representation as words throughout; the element π of S_n is written as the word $1\pi \cdots n\pi$. Let C_n be the set of n-***cycles*** of S_n (recall that an element of S is a k-cycle if and only if its action has a single nontrivial orbit, and this orbit has size k). When $n = 1$, our convention is that $\mathrm{C}_n = \mathrm{S}_n$. A 2-cycle is called a ***transposition***, and we denote by $\tau(i, j)$ the transposition that exchanges i and j and fixes all other symbols.

Finally, for $\pi \in \mathrm{S}_n$, we define $q(\pi) = n\pi^{-1}$, and $Q(\pi) = n\pi = q(\pi^{-1})$.

Random generation. Pseudocode for the two algorithms discussed below is shown in Figure 1. Note that for notational convenience we will consider permutations on $\{0, \ldots, n-1\}$ instead of $\{1, \ldots, n\}$ in that figure, as well as in section 1.

The standard algorithm [Knut1969, 3.4.2, Algorithm P] for uniformly generating a random permutation of $[n]$ is as follows. Start with the identity permutation. There are $n-1$ steps. At the ith step, a random position j is chosen uniformly from $[n-i+1]$ and the current element in position j is swapped with the element at position $n-i+1$. Example: the permutation $25314 \in \mathrm{S}_5$ is formed by choosing $j = 4, 1, 3, 1$ in that order. Knuth attributes this algorithm to R. A. Fisher and F. Yates [FY1938], and a computer implementation was given by Durstenfeld [Durs1964]; it is often called the "Fisher-Yates shuffle" or the "Knuth shuffle".

S. Sattolo [Satt1986] introduced a very similar algorithm for uniform random generation of an element of C_n. The only difference in the algorithm is that the possibility $j = n-i+1$ is disallowed; j is chosen uniformly at random from $[n-i]$.

2000 *Mathematics Subject Classification.* 68W20, 68W40, 68Q25, 05A05.

Key words and phrases. Sattolo's algorithm, Mahonian permutation statistic.

Thanks to Hosam Mahmoud and Philippe Flajolet for useful discussions.

algorithm `fisher-yates`
Input: positive integer n
Output: permutation $\pi \in \mathrm{S}_n$
begin
for i **from** 0 **to** $n-1$ **do**
$\pi[i] := i$
if $(n > 1)$ **then**
for k **from** $n-1$ **downto** 1 **do**
$j := \mathtt{rand}(0..k)$
$\pi \leftarrow \mathtt{swap}(\pi, r, k)$
return(π)
end

algorithm `sattolo`
Input: positive integer n
Output: permutation $\pi \in \mathrm{C}_n$
begin
for i **from** 0 **to** $n-1$ **do**
$\pi[i] := i$
if $(n > 1)$ **then**
for k **from** $n-1$ **downto** 1 **do**
$j := \mathtt{rand}(0..k-1)$
$\pi \leftarrow \mathtt{swap}(\pi, r, k)$
return(π)
end

Figure 1: The random generation algorithms

Algebraic description. In terms of multiplication in the symmetric group S_n, the description of the Fisher-Yates algorithm is as follows. For each $i, j \in [n]$, let $\tau(i,j)$ denote the transposition that exchanges i and j. Then each algorithm starts with the identity permutation π. At each value of k, the statement $\pi \leftarrow \pi\tau(r\pi, j\pi)$ is executed. In terms of multiplication on the left, we have the following. Observe that the transposition $\tau(r\pi, j\pi)$ equals the conjugation $\pi^{-1}\tau(r,j)\pi$. Thus we can rewrite the step above as $\pi \leftarrow \tau(r,j)\pi$.

Definition 1.1. We call a product $\tau_1 \cdots \tau_k$ of transpositions a ***triangular product*** if for each $i \le k$, τ_i exchanges $i+1$ and some $j \le i+1$. The product is called a ***strict triangular product*** if always $j \le i$.

It follows directly from the above discussion that each execution of the Fisher-Yates (respectively Sattolo's) algorithm yields a triangular (respectively strictly triangular) product of $n-1$ transpositions in S_n. Furthermore these maps are 1–1. To see this, note that given π we can reconstruct τ_i for each i. This is because τ_i fixes all elements greater than $i+1$, so that $n \cdot \tau_{n-1} = n \cdot \pi$. This determines the transposition τ_{n-1}, and the result follows by induction on n.

We may therefore define maps $^{\uparrow} : \pi_{n-1} \times [n-1] \to \pi_n$ and $(_{\downarrow}, q) : \pi_n \to \pi_{n-1} \times [n-1]$ by

$$(\tau_1 \dots \tau_{n-2}, q)^{\uparrow} = \tau(q,n)\tau_1 \dots \tau_{n-2} = \tau_1 \dots \tau_{n-2}\tau(n, q \cdot \tau_{q-1})$$
$$(\tau_1 \dots \tau_{n-1})_{\downarrow} = \tau_1 \dots \tau_{n-2}$$

Note that $(\pi^{-1})_{\downarrow} = (\pi_{\downarrow})^{-1}$. Note also that we could also define $^{\uparrow}$ and $_{\downarrow}$ directly without reference to the triangular representation:

$$(\pi, q)^{\uparrow} = \tau(n(\pi)+1, q)\pi^* = \pi^*\tau(n(\pi)+1, q \cdot \pi))$$
$$\pi_{\downarrow} = (\tau(n(\pi), q(\pi))\pi)_* = (\pi\tau(n(\pi), Q(\pi)))_*.$$

Proposition 1.1. *For $n \ge 2$, the maps $^{\uparrow} : \pi_{n-1} \times [n-1] \to \pi_n$ and $(_{\downarrow}, q) : \pi_n \to \pi_{n-1} \times [n-1]$ defined above are mutually inverse bijections. Furthermore each restricts to* C *and the restrictions are also mutually inverse bijections.*

Proof. Note that if π' denotes $\tau(n(\pi)+1, q)\pi^*$, then $q(\pi') = q$ and $n(\pi') = n(\pi)+1$. Thus the composition of the maps in either order is the identity. Suppose that $\sigma \in \mathrm{C}_{n-1}$ and $1 \le q < n-1$. Then $\rho := \sigma^{\uparrow}$ has the property that $n \cdot \rho^i = q \cdot \sigma^i$ for $1 \le i \le n-1$, and hence never equals n. Thus ρ is an n-cycle. □

By iteration this yields a map $\mathrm{iso}_n : \mathrm{S}_n \to [n]! := [n] \times [n-1] \times \cdots \times [2] \times [1]$ taking a permutation to the sequence of positions j made in the execution of the Fisher-Yates algorithm. Sattolo's algorithm fits nicely into this picture. For each i, there is a natural inclusion map $[i] \to [i+1]$. The product of these gives a map $\iota_{n-1} : [n-1]! \to [n]!$.

We summarize the above result in a proposition.

Proposition 1.2. *The following sets are in bijection via the correspondences described above.*

1. *the set of possible outputs of the Fisher-Yates algorithm;*
2. S_n*;*
3. *the set of triangular decompositions of length* $n-1$ *in* S_n*;*
4. *the set* $[n] \times [n-1] \times \cdots \times [2] \times [1]$.

Furthermore, the measure induced on S_n *by the Fisher-Yates algorithm is uniform.*

The following sets are in bijection via the correspondences described above.

1. *the set of possible outputs of Sattolo's algorithm;*
2. C_n*;*
3. *the set of strict triangular decompositions of length* $n-1$ *in* S_n*;*
4. *the set* $[n-1] \times [n-2] \times \cdots \times [2] \times [1]$.

Furthermore, the measure induced on C_n *by Sattolo's algorithm is uniform.*

Note that the initial subproduct of length i of a (strict) triangular product $\tau_1 \dots \tau_{n-1}$ of transpositions in S_n is itself a (strict) triangular product of transpositions in S_{i+1}, and hence an $(i+1)$-permutation/cycle. At each stage, forming the next partial product involves inserting $i+1$ into the current permutation/cycle. This gives an algorithm for forming a uniformly random permutation/cycle of a random length; simply form such a (strict) triangular product with length chosen according to the desired distribution. Clearly the distribution conditioned on the length is uniform.

Exhaustive generation. There are obvious deterministic versions of the above algorithm. Instead of randomly choosing the transpositions, we simply run through all such transpositions systematically. Every method of generating all elements of $[n]!$ (and the corresponding unranking function) can be transferred via the encoding above to a method for generating all elements of S_n or C_{n+1}.

A common way of enumerating a combinatorial class is to use an ***incremental method***, where each object is generated from the last using a small change. The standard minimal-change algorithms for permutation generation are given in [Knut2004, 7.2.1.2]. A very general method of enumerating permutations is as follows. A ***Sims table*** for a subgroup G of S_n is a family of subsets $S_1, \dots$ of G having the following property: for each j, k with $1 \le j \le k \le n$, S_k contains exactly one element that fixes all elements greater than k and takes k to j, whenever G itself contains such a permutation. It is easily seen [Knut2004, Lemma S] that if $S_1, \dots, S_{n-1}$ is a Sims table then every element of G has a unique representation as a product $\pi = \pi_1 \dots \pi_{n-1}$, where $\pi_k \in S_k$ for each k. There is also a unique dual representation of the form $\pi = \pi_{n-1}^{-1} \dots \pi_1^{-1}$ with $\pi_k \in S_k$, obtained by inverting the first representation for π^{-1}.

An inspection of the proofs shows that G need not be a group for such results to hold. In fact, it is only necessary that G be closed under taking inverses. Thus, for example, the set C_n could be used.

The triangular decomposition fits into this framework. For each k, the set S_k consists of all transpositions τ_{jk} with $j \le k$ (in the case S_n) or $j < k$ (in the case C_n). The Sims representation with respect to these sets S_k is precisely the triangular representation.

For comparison we include the usual inversion encoding in Table 3.

Lex order on $[n] \times \cdots \times [1]$				Induced order on S_n			
0000	0100	0200	0300	1230	3201	1302	1203
0001	0101	0201	0301	2130	2301	3102	2103
0010	0110	0210	0310	2310	2031	3012	2013
0011	0111	0211	0311	3210	0231	0312	0213
0020	0120	0220	0320	1320	3021	1032	1023
0021	0121	0221	0321	3120	0321	0132	0123

Table 1: The Fisher-Yates encoding for $n = 4$.

Lex order on $[1] \times \cdots \times [n]$						Induced order on S_n					
0000	0010	0020	0100	0110	0120	1230	2310	1320	2130	3210	3120
0001	0011	0021	0101	0111	0121	3201	2031	3021	2301	0231	0321
0002	0012	0022	0102	0112	0122	1302	3012	1032	3102	0312	0132
0003	0013	0023	0103	0113	0123	1203	2013	1023	2103	0213	0123

Table 2: The Fisher-Yates encoding for $n = 4$.

Lex order on inversion function						Induced order on S_n					
0000	0010	0020	0100	0110	0120	0123	0213	2013	1023	1203	2103
0001	0011	0021	0101	0111	0121	0132	0231	2031	1032	1230	2130
0002	0012	0022	0102	0112	0122	0312	0321	2301	1302	1320	2310
0003	0013	0023	0103	0113	0123	3012	3021	3201	3102	3120	3210

Table 3: The inversion encoding for $n = 4$.

A ***Gray code*** for $G = S_n$ is a Hamiltonian path in the Cayley graph of G where the generating set is the set of all transpositions. The usual Gray code on words in $[1] \times \cdots \times [n]$ induces a Gray code on G via the inversion encoding, since each minimal change to a word corresponds to a transposition of adjacent symbols. This can be seen by reading the columns alternately downwards and upwards from left to right in Table 3.

What happens when we use instead the Fisher-Yates encoding? The Gray code order on words induces a Hamiltonian path in the Cayley graph of G, but with respect to a different set of generators. The generators in question are in fact transpositions and 3-cycles. To see this, note that to get from one entry to the next we move from $\pi_1\tau\pi_2$ to $\pi_1\tau'\pi_2$, which is achieved by multiplying by $\pi_2^{-1}\tau'\tau\pi_2$. Since τ and τ' transpose some symbol k with j, j' respectively, where $j \neq j'$ and $j, j' \leq k$, the product $\tau'\tau$ is the permutation that moves j' to j, j to k and k to j'. If either $j = k$ or $j' = k$ then $\tau'\tau$ is a transposition, and otherwise it is a 3-cycle. The conjugation by π_2 preserves the cycle structure.

Note that the restriction to C is better behaved and the Cayley graph of the set C with respect to the set of 3-cycles has a Hamiltonian cycle. Since we always have $j, j' < k$ in the case of C, the transpositions are never needed and we always move from one element to the next by multiplying by a 3-cycle. Since smaller changes could only be transpositions, and multiplying an n-cycle by a transposition can never yield an n-cycle, the enumeration described above deserves the name "***Gray code for cycles***".

2. Analysis of some quantities

Obvious quantities to be studied are: the number of swaps; the number of times a given symbol is chosen by the random calls (we call this the number of moves, although some of these moves will be trivial); the total distance moved by a given symbol; the total distance moved. The second and third of these were discussed in [Prod2002, Mahm2003, Wils2004] for the case of Sattolo's algorithm.

The number of swaps is always $n-1$ for each algorithm, but some of these can be trivial (and hence executed more quickly) for the Fisher-Yates algorithm, whereas every exchange is nontrivial in Sattolo's algorithm. The number of nontrival swaps is the number of elements moved by the permutation, or $n-f$, where f is the number of fixed points. The generating function for permutations by size and fixed points is well known to be

$$\sum_{\pi} \frac{x^{|\pi|}}{|\pi|!} u^{f(\pi)} = \frac{e^{(u-1)z}}{1-z}.$$

For example, the expected number of fixed points is 1 for every n.

Number of moves and distance moved by an element. To avoid excessive case distinctions we consider the slight variant of these algorithms in which the final "swap" of $\pi[0]$ with itself is performed (this corresponds to the "downto" loops in Figure 1 going down to 0 instead of 1.

We consider normalized counting generating functions of the form

$$F(u,t,x) := \sum_{\pi\in \mathrm{S}, p\in[n(\pi)]} u^{\chi(\pi,p)} t^p \frac{x^{n(\pi)}}{|\,\mathrm{S}_{n(\pi)}\,|} = \sum_{n\geq 1} \frac{x^n}{n!} \sum_{1\leq p\leq n} t^p \sum_{\pi\in \mathrm{S}_n} u^{\chi(\pi,p)}.$$

An auxiliary "diagonal" GF will also be useful:

$$G(u,x) := \sum_{\pi\in \mathrm{S}} u^{\chi(\pi,n(\pi))} \frac{x^{n(\pi)}}{|\,\mathrm{S}_{n(\pi)}\,|} = \sum_{n\geq 1} \frac{x^n}{n!} \sum_{\pi\in \mathrm{S}_n} u^{\chi(\pi,n)}.$$

Here χ is a given parameter of interest such as number of moves, etc. Of course F and G can be interpreted probabilistically as "grand" PGFs. For example, if $\chi(\pi,p)$ is the number of moves made by p in obtaining π via the Fisher-Yates algorithm, and M_{np} the random variable obtained by evaluating χ at an element of C_n chosen uniformly at random, then letting $\phi_{np}(u) = \sum_{l\geq 0} \mathbb{P}(M_{np}=l)u^l$ denote the PGF of M_{np}, we have

$$F(u,t,x) = \sum_{n\geq 1} x^n \sum_{p=1}^{n} t^p \phi_{np}(u).$$

We first consider the case where χ is the number of moves of a given symbol. The triangular decomposition yields the recurrence

$$\chi(\pi,p) = \begin{cases} \chi(\pi_\downarrow, p) & \text{if } p\neq n(\pi), p\neq q(\pi); \\ 1+\chi(\pi_\downarrow, q(\pi)) & \text{if } p=n(\pi), p\neq q(\pi); \\ 1 & \text{if } p\neq n(\pi), p=q(\pi); \\ 1 & \text{if } p=n(\pi), p=q(\pi). \end{cases} \tag{2.1}$$

In the case where χ is the distance moved by an element, we have the recurrence

$$\chi(\pi,p) = \begin{cases} \chi(\pi_\downarrow, p) & \text{if } p\neq n(\pi), p\neq q(\pi); \\ n(\pi)-q(\pi)+\chi(\pi_\downarrow, q(\pi)) & \text{if } p=n(\pi), p\neq q(\pi); \\ n(\pi)-q(\pi) & \text{if } p\neq n(\pi), p=q(\pi); \\ 0 & \text{if } p=n(\pi), p=q(\pi). \end{cases} \tag{2.2}$$

We partition the index set $\mathcal{I} = \{(\pi, p) \mid \pi \in \mathrm{S}, 1 \le p \le n(\pi)\}$ into 4 disjoint subsets $\mathcal{I}_1, \ldots, \mathcal{I}_4$ according to the cases just listed. Denote by $\Sigma_k(u,t,x)$ the part of the sum defining F corresponding to index set $\mathcal{I}_k$, so that $F = \Sigma_1 + \Sigma_2 + \Sigma_3 + \Sigma_4$.

Note that for each χ we have

$$\begin{aligned}\Sigma_4 + \Sigma_2 &= \sum_{n\ge1} \frac{x^n}{n!} t^n \sum_{\pi\in \mathrm{S}_n} u^{\chi(\pi,n)} \\ &= G(u, tx).\end{aligned}$$

In the sum Σ_1, indices $p \in [n(\pi)]$ satisfying the conditions $p \ne n(\pi), p \ne q(\pi)$ occur if and only if $n(\pi) \ge 2$. The set $\mathcal{I}_1$ is in bijection with the set

$$\{(\pi, q, p) \mid n(\pi) \ge 2, 1 \le p < n(\pi), 1 \le q \le n(\pi), p \ne q\}.$$

Let $A(u,t,x)$ be the antiderivative of $F(u,t,x)$ with respect to x having $A(u,t,0) = 0$. Then for each χ we obtain

$$\begin{aligned}\Sigma_1(u,t,x) &= \sum_{n\ge2} \frac{x^n}{n!} \sum_{\pi\in \mathrm{S}_n} \sum_{1\le p<n, p\ne q(\pi)} t^p u^{\chi(\pi,p)} \\ &= \sum_{n\ge2} \frac{x^n}{n!} \sum_{\pi\in \mathrm{S}_{n-1}} \sum_{1\le q\le n} \sum_{1\le p<n, p\ne q} u^{\chi(\pi,p)} t^p \\ &= x \sum_{n\ge1} \frac{x^n}{(n+1)!} \sum_{1\le p\le n} t^p \sum_{\pi\in \mathrm{S}_n} u^{\chi(\pi,p)} \sum_{1\le q\le n+1, q\ne p} 1 \\ &= x \sum_{n\ge1} \frac{nx^n}{(n+1)!} \sum_{1\le p\le n} t^p \sum_{\pi\in \mathrm{S}_n} u^{\chi(\pi,p)} \\ &= x \sum_{n\ge1} \frac{x^n}{n!} \sum_{1\le p\le n} t^p \sum_{\pi\in \mathrm{S}_n} u^{\chi(\pi,p)} - \sum_{n\ge1} \frac{x^{n+1}}{(n+1)!} \sum_{1\le p\le n} t^p \sum_{\pi\in \mathrm{S}_n} u^{\chi(\pi,p)} \\ &= xF(u,t,x) - A(u,t,x).\end{aligned}$$

We now determine $\Sigma_3(u,t,x)$. The set $\mathcal{I}_3$ is in bijection with $\{\pi \in \mathrm{S} \mid n(\pi) \ge 2, q \ne n(\pi)\}$. Thus for the number of moves we have

$$\begin{aligned}\Sigma_3(u,t,x) &= \sum_{n\ge2} \frac{x^n}{n!} \sum_{\pi\in \mathrm{S}_n} t^{q(\pi)} u^1 = u \sum_{n\ge2} \frac{x^n}{n!} \sum_{\pi_\downarrow \in \mathrm{S}_{n-1}} \sum_{q=1}^{n} t^q \\ &= u \sum_{n\ge1} \frac{x^{n+1}}{(n+1)!} \sum_{\pi\in \mathrm{S}_n} \sum_{q=1}^{n} t^q = u \sum_{n\ge1} \frac{x^{n+1}}{n+1} \sum_{q=1}^{n} t^q \\ &= \frac{utx}{1-t} \sum_{n\ge1} \frac{x^n(1-t^n)}{n+1} = \frac{u}{1-t}[\log(1-tx) - t\log(1-x)].\end{aligned}$$

Note that when $t = 1$ we have the formula $\Sigma_3(u,1,x) = u\log(1-x) + ux/(1-x)$.

To obtain Σ_3 for the distance moved, a similar calculation yields

$$\frac{u}{u-t}\left[u\log(1-tx) - t\log(1-ux)\right].$$

We now consider G. We have for the number of moves

$$\begin{aligned}
G(u,x) &= \sum_{n\geq 1}\frac{x^n}{n!}\sum_{\pi\in \mathrm{S}_n} u^{\chi(\pi,n)} = \sum_{n\geq 2}\frac{x^n}{n!}\sum_{\pi\in \mathrm{S}_n, q(\pi)\neq n(\pi)} u^{\chi(\pi,n)} + \sum_{n\geq 1}\frac{x^n}{n!}\sum_{\pi\in \mathrm{S}_n, q(\pi)= n(\pi)} u^{\chi(\pi,n)} \\
&= \sum_{(\pi_\downarrow, 1\leq q\leq n(\pi_\downarrow))} u^{1+\chi(\pi_\downarrow,q)}\frac{x^{n(\pi_\downarrow)+1}}{n(\pi_\downarrow+1)!} + \sum_{n\geq 1}\frac{x^n}{n!}\sum_{\pi\in \mathrm{S}_n, q(\pi)=n(\pi)} u^1 \\
&= u\sum_{(\pi, 1\leq q\leq n(\pi))} u^{\chi(\pi,q)}\frac{x^{n(\pi)+1}}{(n(\pi)+1)!} + u\sum_{n\geq 1}\frac{x^n}{n!}(n-1)! \\
&= uA(u,1,x) - u\log(1-x).
\end{aligned}$$

Similarly in the case of distance moved we obtain $G(u,x) = A(u,u^{-1},ux) - \log(1-x)$.

Thus for the number of moves, by differentiating we obtain the system

$$\begin{aligned}
(1-x)F'(u,t,x) &= tG'(u,tx) + \Sigma_3'(u,t,x) \\
G'(u,x) &= uF(u,1,x) + \frac{u}{1-x}.
\end{aligned}$$

Substituting $t=1$ and eliminating $G'(u,x)$ we obtain

$$\begin{aligned}
(1-x)F'(u,1,x) - uF(u,1,x) = \frac{u}{1-x} + \Sigma_3'(u,1,x) &= \frac{u}{(1-x)^2} \\
F(u,1,0) &= 0
\end{aligned}$$

which yields

$$F(u,1,x) = \frac{u}{2-u}\left[(1-x)^{-2} - (1-x)^{-u}\right].$$

From this G can be found explicitly via a single integration.

$$G(u,x) = \frac{u^2}{2-u}\left[\frac{1}{1-x} + \frac{(1-x)^{1-u}}{1-u}\right] - \frac{u^2}{1-u} - u\log(1-x).$$

To find F explicitly is more difficult, because it requires the integration of $(1-tx)^{-u}(1-x)^{-1}$, and we do not pursue it here. In any case we have the defining equation

$$\begin{aligned}
(1-x)F'(u,t,x) &= utF(u,1,tx) + \frac{u}{1-tx} + \Sigma_3'(u,t,x) \\
F(u,t,0) &= 0.
\end{aligned}$$

Using this defining equation we may easily extract the coefficient of $x^n t^p$ to obtain the probability generating function $\phi_{np}(u)$, or extract moments by evaluating appropriate partial u-derivatives at $u=1$ as usual. For example, $\phi_{np}(u)$ is the coefficient of $x^n t^p$ in F, hence equals $(1/n)$ times the coefficient of $x^n t^p$ in xF', and the mean of the random variable with PGF $\phi_{np}(u)$ can therefore be obtained by evaluating $x\partial^2 F/\partial x\partial u$ at $u=1$, then dividing by n. The defining equation allows us to express these derivatives in terms of derivatives of the known series $G(u,x)$ and $F(u,1,x)$.

In detail, we see that (with subscripts denoting partial derivatives, and $\Sigma = \Sigma_3$ to avoid notational overload)

$$\begin{aligned}
x(1-x)F_{13}(1,t,x) &= txG_{12}(1,tx) + x\Sigma_{13}(1,t,x) \\
&= \frac{tx}{(1-tx)^2} + \frac{2t^2x^2}{(1-tx)^2} + \frac{tx\log(1-tx)}{1-tx} + \frac{tx}{(1-x)(1-tx)}.
\end{aligned}$$

Extracting the coefficient of t^p from the right side yields $px^p + 2(p-1)x^p + x^{p+1}/(1-x) - H_{p-1}$ where H_p denotes the pth harmonic number $\sum_{1\le i\le p} 1/i$. Dividing by $(1-x)$ and extracting the coefficient of x^n yields (where M denotes the number of moves)

$$E[M_{np}] = \frac{n + 2p - 2 - H_{p-1}}{n}.$$

Higher moments can also be obtained with more calculation of the same type, but we do not pursue this aspect here.

We can also immediately extract recurrences for the probability generating function. We obtain

$$\begin{aligned}\phi_{np}(u) &= \frac{p}{n}\phi_{pp}(u) + (1-p/n)u \\ \phi_{nn}(u) &= \frac{u}{n}\left[1 + \sum_{1\le p\le n-1} \phi_{n-1,p}(u)\right].\end{aligned}$$

We can now easily write down an explicit formula for the probability generating function ϕ_{np} by extracting of coefficients from G and using the first recurrence above. We have

$$\begin{aligned}\phi_{nn}(u) &= \frac{u}{n} + \frac{u^2}{2-u}\left[1 - \frac{u(u+1)\dots(u+n-2)}{n!}\right] \\ \phi_{np}(u) &= \frac{n-(p-1)}{n}u + \frac{p}{n}\frac{u^2}{2-u}\left[1 - \frac{u(u+1)\dots(u+p-2)}{p!}\right].\end{aligned}$$

Similarly for the distance moved we obtain

$$\begin{aligned}(1-x)F'(u,t,x) &= tG'(u,tx) + \Sigma_3'(u,t,x) \\ G'(u,x) &= uF(u,u^{-1},ux) + (1-x)^{-1}\end{aligned}$$

which leads via the substitution $t \leftarrow u^{-1}$ to

$$\begin{aligned}(1-x)F'(u,u^{-1},x) - F(u,u^{-1},x) &= (1-u^{-1}x)^{-1} + \Sigma_3'(u,u^{-1},x) \\ F(u,u^{-1},0) &= 0.\end{aligned}$$

This equation is exact and leads to

$$\begin{aligned}F(u,u^{-1},ux) &= \frac{1}{1-ux}\left[-u\log(1-x) + \Sigma_3(u,u^{-1},ux)\right] \\ &= \frac{1}{1-ux}\left[-u\log(1-x) + \frac{u}{1-u^2}\left[\log(1-u^2x) - u^2\log(1-x)\right]\right].\end{aligned}$$

Another integration yields $G(u,x)$ and again we have a defining equation for $F(u,t,x)$. Each of these integrations takes us outside the realm of elementary functions. However, we can systematically extract coefficients as before from the equations that we have. We omit any details of the calculations. The probability generating function ξ_{np} of the random variable D_{np} satisfies the recurrence

$$\begin{aligned}\xi_{np}(u) &= \frac{p}{n}\xi_{pp}(u) + \frac{1}{n}\sum_{j=1}^{n-p} u^j \\ \xi_{nn}(u) &= \frac{1}{n}\left[1 + \sum_{p<n} u^{n-p}\xi_{n-1,p}(u)\right].\end{aligned}$$

They have explicit formulae such as

$$\xi_{nn}(u) = \frac{1}{n}\left[1 + \sum_{j=1}^{n-1} \frac{u^{n-j}}{j} + \frac{u^{n+2}}{1-u^2} \sum_{j=1}^{n-1} \frac{u^{1-j} - u^{j-1}}{j}\right].$$

Similarly we could extract the mean and higher moments as before. From the explicit forms one could consider limit distributions by considering the pointwise limit of the probability generating function ϕ_{np} or ξ_{np}. We do not pursue this further here as the computations are routine but tedious.

The total distance moved. We consider the total distance moved rightwards by elements, $D_n = \sum_p D_{np}$ (this of course equals the total distance moved leftwards by elements). As a random variable, D_n is the sum of D_{n-1} and a random variable U_i that is uniform on $[0..n-1]$. Thus D_n is distributed as $\sum_{i=2}^{n} U_i$ and has probability generating function $\prod_{i=1}^{n} \frac{1-u^i}{1-u}$.

Note that this PGF is the same as the one for inversions. Thus the number of inversions and the total rightward distance have the same distribution (in other words, D_n is a ***Mahonian statistic***). Hence for each k, the number of permutations in S_n with k inversions is the same as the number of permutations in S_n whose rightward distance is k.

3. Extensions and discussion

Despite an extensive literature search, I can only find two places in the literature in which the very natural Fisher-Yates encoding is mentioned. In neither paper was the connection with cyclic permutations mentioned.

What we have called the Fisher-Yates encoding was used in [MaRa2001] to study anti-excedances. In [MyRu2001], two unranking and ranking functions for permutations were presented, each taking linear time to compute. Although not mentioned in that paper, it is easily seen that those orderings correspond via the Fisher-Yates encoding to lexicographic order on the triangular cartesian product $[1] \times \cdots \times [n]$ or $[n] \times \cdots \times [1]$ as we have described above.

We note that Sattolo's algorithm is a special case of a method to uniform generation of permutations with a fixed number k of cycles [Wilf].

Suppose that $m \geq 0$ is fixed, and at the ith step, we swap $\pi(i)$ and $\pi(j)$ where j is chosen uniformly from $[1..n+1-i-m]$. The cases $m = 0$ and $m = 1$ respectively correspond to the Fisher-Yates and Sattolo algorithms. Other values of m do not appear to be particularly interesting, although we have not pursued them.

I have not yet been able to determine whether the statistic D above is known. An interesting question is its correlation with other well-known permutation statistics.

There is a small connection between the Fisher-Yates algorithm and sorting.

Proposition 3.1. *Let $\pi \in \mathrm{S}_n$ and let $\tau_1 \ldots \tau_{n-1}$ be its triangular decomposition. Then in order to sort π^{-1}, selection sort applies the transpositions $\tau_{n-1}, \ldots, \tau_1$ in that order.*

Proof. Selection sort first chooses n and puts it in the correct position; this corresponds to postmultiplication by $\tau(n, n\pi)$, which corresponds to premultiplication by $\tau(n, n\pi^{-1})$. The result follows by induction. □

For example, to generate $\pi = 3421$ the algorithm proceeds as follows: 1234, 4231, 4321, 3421, yielding the strict triangular decomposition (12)(23)(14). Applying these in turn to π^{-1} gives $4312, 2314, 2134, 1234$ which is the list created by selection sort when sorting $\pi^{-1} = 4312$.

References

[Durs1964] Richard Durstenfeld, *Algorithm 235: Random permutation*, Comm. Assoc. Comput. Mach. 7, 1964, 420.

[FY1938] R. A. Fisher and F. Yates, Example 12, *Statistical Tables*, London, 1938.

[GX1988] David Gries and Jin Yun Xue, *Generating a random cyclic permutation*, BIT **28** (1988), 569–572.

[Knut1969] Donald E. Knuth, *The art of computer programming. Vol. 2: Seminumerical algorithms.* Addison-Wesley, 1969.

[Knut2004] Donald E. Knuth, *The art of computer programming. Vol. 4, Fasc. 2. Generating all tuples and permutations.* Addison-Wesley, 2005.

[MaRa2001] Roberto Mantaci and Fanja Rakotondrajao, *A permutation representation that knows what "Eulerian" means*, Discrete Math. Theor. Comput. Sci. **4** (2001), 101–108.

[Mahm2003] Hosam M. Mahmoud, *Mixed distributions in Sattolo's algorithm for cyclic permutations via randomization and derandomization*, J. Appl. Probab. **40** (2003), 790–796.

[MyRu2001] Wendy Myrvold and Frank Ruskey, *Ranking and unranking permutations in linear time*, Inform. Process. Lett. 79 (2001), 281–284.

[Prod2002] Helmut Prodinger, *On the analysis of an algorithm to generate a random cyclic permutation*, Ars Combin. **65** (2002), 75–78.

[Prod] ———, Online document at `http://math.sun.ac.za/~prodinger/abstract/abs_161.htm`.

[Satt1986] Sandra Sattolo, *An algorithm to generate a random cyclic permutation*, Inform. Process. Lett.**22** (1986), 315–317.

[Wilf] Herbert Wilf, *East Side, West Side*, lecture notes available from `http://www.cis.upenn.edu/~wilf/lecnotes.html`.

[Wils2004] Mark C. Wilson, *Probability generating functions for Sattolo's algorithm*, J. Iranian Stat. Soc. 3 (2004), 297–308.

Department of Computer Science, University of Auckland, Private Bag 92019 Auckland, New Zealand

E-mail address: `mcw@cs.auckland.ac.nz`