Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 140 (2004) 207–221
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Gray code for derangements

Jean-Luc Baril, Vincent Vajnovszki*
LE2I-UMR CNRS 5158, Université de Bourgogne, B.P. 47 870, 21078 Dijon Cedex, France

Received 24 May 2002; received in revised form 27 May 2003; accepted 7 June 2003

Abstract

We give a Gray code and constant average time generating algorithm for *derangements*, i.e., permutations with no fixed points. In our Gray code, each derangement is transformed into its successor either via one or two transpositions or a rotation of three elements. We generalize these results to permutations with number of fixed points bounded between two constants.

Keywords: Restricted permutations; Gray codes; Generating algorithms

1. Introduction

Various studies have been made on Gray codes and generating algorithms for permutations and their restrictions (with given *ups* and *downs* [9,11] or inversions [6,17], involutions, and fixed-point free involutions [18]) or their generalizations (multiset permutations [8,16]). See [7,13] for surveys of permutation generation methods.

A length-n *derangement* (or *rencontre* or *coincidence*) is a permutation $\pi \in S_n$ with no fixed points, i.e., $\pi(i) \neq i$ for all $i \in [n] = \{1, 2, \ldots, n\}$. If D_n is the set of all length-n derangements, then a recurrence relation for $d_n = card(D_n)$ is given by

$$d_n = (n-1)(d_{n-1} + d_{n-2}) \tag{1}$$

for $n \geqslant 2$, with $d_1 = 0$ and $d_2 = 1$; see for instance [4, p. 180] or [14, p. 67]. There are sequential [1] and parallel [2, p. 650] algorithms for generating derangements in lexicographic order. However, we know of no published algorithms for derangements in Gray code order. Here, we present such an algorithm which is based on the combinatorial proof of relation (1) above.

* Corresponding author.
E-mail addresses: barjl@u-bourgogne.fr (J-L. Baril), vvajnov@u-bourgogne.fr (V. Vajnovszki).

doi:10.1016/j.dam.2003.06.002

We represent permutations in one-line notation; i.e., $\pi=(i_1,i_2,\dots,i_n)$ iff $\pi(k)=i_k$, and if $\sigma=(\sigma_1,\sigma_2,\dots,\sigma_n)$ is a length-n integer sequence, then $\sigma\cdot\pi$ is the sequence $(\sigma_{\pi(1)},\sigma_{\pi(2)},\dots,\sigma_{\pi(n)})$. As a particular case, when $\sigma\in S_n$ then $\sigma\cdot\pi\in S_n$ is their composition (or product).

Let $i_1,i_2,\dots,i_k$ be k different values in $[n]=\{1,2,\dots,n\}$, $1\leqslant k\leqslant n$. The *cycle* $\gamma=\langle i_1,i_2,\dots,i_k\rangle$ is the following permutation: $\gamma(i_1)=i_2$, $\gamma(i_2)=i_3,\dots,\gamma(i_{k-1})=i_k$, $\gamma(i_k)=i_1$, and $\gamma(j)=j$ for all $j\neq i_\ell$, $1\leqslant\ell\leqslant k$. A length-two cycle is a *transposition*, and each cycle can be written as a product of transpositions: $\langle i_1,i_2,\dots,i_k\rangle=\langle i_1,i_k\rangle\cdot\langle i_1,i_{k-1}\rangle\cdot\dots\cdot\langle i_1,i_2\rangle$ for $k\geqslant 2$. Also, the composition of two cycles with disjoint domains is commutative, and each permutation is the product of cycles with disjoint domains. In a permutation $\sigma\in S_n$, transposing the *positions* i and j corresponds to the product $\sigma\cdot\langle i,j\rangle$ and transposing the *values* x and y corresponds to the product $\langle x,y\rangle\cdot\sigma$.

For a length-n integer sequence $\alpha=(\alpha(1),\alpha(2),\dots,\alpha(n))$ and a permutation π in S_n, we say that π is the *normal form* of α if α is *order-isomorphic* to π, i.e., $\alpha(i)<\alpha(j)$ if and only if $\pi(i)<\pi(j)$ for all $1\leqslant i,j\leqslant n$. In this case, all the elements of α are distinct.

In the Gray code we give in the next section, a derangement is obtained from the previous one via one or two transpositions, and, as a particular case when the domains of the two transpositions are not disjoint, via a length-three cycle. In Section 3 this code is implemented as a generating algorithm and in Section 4 it is extended for permutations with a given number of fixed points and for permutations with the number of fixed points between two bounds.

A list $\mathscr{L}$ for a set L of integer sequences is an ordered list of the elements of L. *first*$(\mathscr{L})$ is the first element and *last*$(\mathscr{L})$ the last element on the list $\mathscr{L}$; $\overline{\mathscr{L}}$ is the list obtained by reversing $\mathscr{L}$, and obviously *first*$(\mathscr{L})=$ *last*$(\overline{\mathscr{L}})$ and *first*$(\overline{\mathscr{L}})=$ *last*$(\mathscr{L})$; $\mathscr{L}^{(i)}$ is the list $\mathscr{L}$ if i is even, and $\overline{\mathscr{L}}$ if i is odd; if $\mathscr{L}_1$ and $\mathscr{L}_2$ are two lists, then $\mathscr{L}_1\circ\mathscr{L}_2$ is their concatenation, and generally $\bigcirc_{i=1}^{n}\mathscr{L}_i$ is the list $\mathscr{L}_1\circ\mathscr{L}_2\circ\dots\circ\mathscr{L}_n$.

2. The Gray code

In this section, first we show how the set D_n can by recursively constructed from D_{n-1} and D_{n-2}, and then we extend this construction to lists of derangements in order to obtain a Gray code.

Let τ be a length-$(n-1)$ derangement, $n\geqslant 3$, and let i be an integer such that $1\leqslant i\leqslant n-1$. If we denote by σ the permutation in S_n obtained from τ by replacing the entry with *value* i by n and appending i in the last position, then σ is a length-n derangement with n not belonging to a transposition.

Similarly, let τ be a length-$(n-2)$ derangement, $n\geqslant 4$, and let i be an integer such that $1\leqslant i\leqslant n-1$. If σ denotes the permutation in S_n obtained from τ by: (1) adding one to each entry greater than or equal to i, (2) inserting n in position i, and finally, (3) appending i in the last position, then σ is a length-n derangement with n belonging to a transposition (the transposition $\langle i,n\rangle$). Moreover, each length-n derangement, $n\geqslant 4$, can be uniquely obtained by one of these constructions.

More formally, for $n > 0$, let D'_n be the set of length-n derangements where n does not belong to a transposition and D''_n its complement, i.e., n belongs to a transposition. Clearly, $D'_1 = D''_1 = D'_2 = \emptyset$, and $D_2 = D''_2$ is the single derangement list $(2,1)$; and $D'_n \cup D''_n$ is a two-partition of the set D_n of all length-n derangements. The functions ϕ and ψ defined below give a bijection between $[n-1] \times D_{n-1}$ and D'_n on the one hand and between $[n-1] \times D_{n-2}$ and D''_n on the other.

Definition 1. (1) For $n \geqslant 3$, an integer $i \in [n-1]$ and a derangement $\tau \in D_{n-1}$, we define a length-n permutation $\sigma = \phi_n(i,\tau)$ by

$$\sigma(j) = \begin{cases} n & \text{if } \tau(j) = i, \\ i & \text{if } j = n, \\ \tau(j) & \text{otherwise.} \end{cases}$$

(2) For $n \geqslant 4$, an integer $i \in [n-1]$ and a derangement $\tau \in D_{n-2}$, we define a length-n permutation $\sigma = \psi_n(i,\tau)$ by

$$\sigma(j) = \begin{cases} i & \text{if } j = n, \\ n & \text{if } j = i, \\ \tau(j) & \text{if } j < i \text{ and } \tau(j) < i, \\ \tau(j)+1 & \text{if } j < i \text{ and } \tau(j) \geqslant i, \\ \tau(j-1) & \text{if } j > i \text{ and } \tau(j-1) < i, \\ \tau(j-1)+1 & \text{if } j > i \text{ and } \tau(j-1) \geqslant i. \end{cases}$$

With i and τ as above, it is easy to see that

- $\phi_n(i,\tau) \in D'_n$ and $\phi_n : [n-1] \times D_{n-1} \to D'_n$ is a bijection; and
- $\psi_n(i,\tau) \in D''_n$ and $\psi_n : [n-1] \times D_{n-2} \to D''_n$ is a bijection.

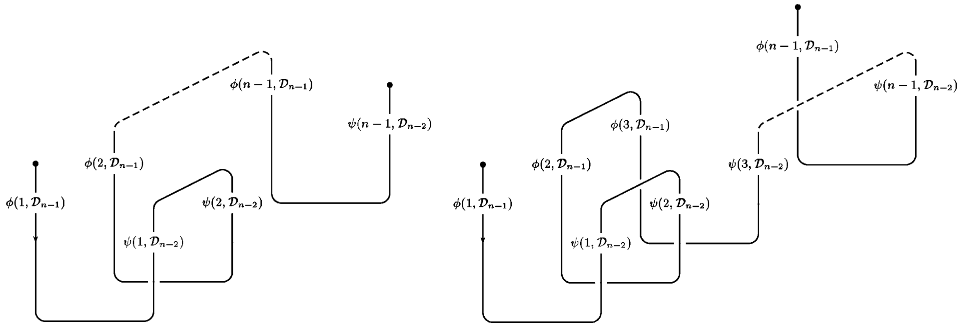
So, for $d_n = card(D_n)$ we have $d_n = card(D'_n) + card(D''_n) = (n-1)d_{n-1} + (n-1)d_{n-2}$ which is a combinatorial proof of (1).

Conversely, we have.

Remark 2. If $\sigma \in D_n$, $n \geqslant 4$, and $i = \sigma(n)$, then

(1) if $\sigma(i) \neq n$ (n is not in a transposition in σ) then $\sigma = \phi_n(i,\tau)$ with τ the permutation represented by the first $n-1$ entries of $\langle i, n \rangle \cdot \sigma$;
(2) if $\sigma(i) = n$ (n is in a transposition in σ) then $\sigma = \psi_n(i,\tau)$ with τ the permutation represented by the normal form of the sequence $(\sigma(1), \sigma(2), \ldots, \sigma(i-1), \sigma(i+1), \ldots, \sigma(n-1))$.

In the following we will omit the subscript n for the functions ϕ and ψ, and it should be clear by context. Also, we extend the functions ϕ and ψ in a natural way to sets and lists of derangements. For $i \in [n-1]$ and $\mathscr{L}$ a list of length-$(n-1)$ derangements we



(a) (b)

Fig. 1. The list $\mathscr{D}_n$: (a) n is even, (b) n is odd.

have $\phi(i,\overline{\mathscr{L}})=\overline{\phi(i,\mathscr{L})}$, $\phi(i, \mathit{first}(\mathscr{L}))=\mathit{first}(\phi(i,\mathscr{L}))$, and $\phi(i, \mathit{last}(\mathscr{L}))=\mathit{last}(\phi(i,\mathscr{L}))$. Similar results hold for the function ψ.

Let $\mathscr{D}_n$ be the list for the set D_n defined by

$$\begin{aligned}
\mathscr{D}_n &= \phi(1,\mathscr{D}_{n-1})\circ\psi(1,\overline{\mathscr{D}}_{n-2}),\\
&\quad \circ\psi(2,\mathscr{D}_{n-2})\circ\phi(2,\overline{\mathscr{D}}_{n-1}),\\
&\quad \circ\phi(3,\mathscr{D}_{n-1})\circ\psi(3,\overline{\mathscr{D}}_{n-2}),\\
&\quad \vdots\\
&= \bigcirc_{i=1}^{n-1}(\phi(i,\mathscr{D}_{n-1})\circ\psi(i,\overline{\mathscr{D}}_{n-2}))^{(i+1)}
\end{aligned} \tag{2}$$

for $n \geqslant 3$, anchored by $\mathscr{D}_1=\psi(1,\emptyset)=\psi(2,\emptyset)=\emptyset$ and $\mathscr{D}_2=(2,1)$.

In Fig. 1 below, the list $\mathscr{D}_n$ is illustrated for even and odd n by a path, where going down means generating a sublist in direct order and going up means generating it in reverse order.

Let f_n denote the first derangement in the list $\mathscr{D}_n$ and ℓ_n denote the last one. The following lemma evaluates f_n and ℓ_n for all n.

Lemma 3. *If $n \geqslant 3$ then*

(1) $f_n=(2,3,\dots,n-1,n,1)$;

(2) $\ell_n=\begin{cases}(2,3,\dots,n-2,n,1,n-1) & \textit{if } n \textit{ is odd},\\ (2,3,\dots,n-2,1,n,n-1) & \textit{if } n \textit{ is even}.\end{cases}$

Proof. (1) $f_n=\phi(1,f_{n-1})$, and by the induction hypothesis, $f_n=\phi(1,(2,3,\dots,n-1,1))=(2,3,\dots,n-1,n,1)$.

(2) If n is odd, then

$$\begin{aligned}\ell_n &= last(\overline{\phi(n-1,\mathscr{D}_{n-1}) \circ \psi(n-1,\overline{\mathscr{D}_{n-2}})})\\ &= first(\phi(n-1,\mathscr{D}_{n-1}))\\ &= \phi(n-1,f_{n-1})\\ &= (2,3,\ldots,n-2,n,1,n-1).\end{aligned}$$

If n is even, then

$$\begin{aligned}\ell_n &= last(\phi(n-1,\mathscr{D}_{n-1}) \circ \psi(n-1,\overline{\mathscr{D}_{n-2}}))\\ &= last(\psi(n-1,\overline{\mathscr{D}_{n-2}}))\\ &= \psi(n-1,f_{n-2})\\ &= (2,3,\ldots,n-2,1,n,n-1). \qquad \square\end{aligned}$$

Note that $f_n(j) = \ell_n(j) = j+1$ for all $j = 1,2,\ldots,n-3$.

The next lemma ensures a smooth transition between the sublists in relation (2), namely between: (i) the list $\psi(i,\overline{\mathscr{D}_{n-2}})$ and $\psi(i+1,\mathscr{D}_{n-2})$, with i odd; (ii) the list $\phi(i,\overline{\mathscr{D}_{n-1}})$ and $\phi(i+1,\mathscr{D}_{n-1})$ with i even; and (iii) the list $\phi(i,\mathscr{D}_{n-1})$ and $\psi(i,\overline{\mathscr{D}_{n-2}})$, or equivalently, the list $\psi(i,\mathscr{D}_{n-2})$ and $\phi(i,\overline{\mathscr{D}_{n-1}})$. More precisely, successive derangements in $\mathscr{D}_n$ differ either by one or two transpositions or by a circular shift of three elements (See Table 1 for two examples).

Lemma 4. (i) *If $n \geqslant 4$ then*

$$\psi(i+1,f_{n-2}) = \begin{cases} \psi(i,f_{n-2}) \cdot \langle i-1,n\rangle \cdot \langle i,i+1\rangle & \text{if } 1 < i \leqslant n-2,\\ \psi(i,f_{n-2}) \cdot \langle 1,2\rangle \cdot \langle n-1,n\rangle & \text{if } i = 1\end{cases} \tag{3}$$

Table 1
The lists $\mathscr{D}_4$ and $\mathscr{D}_5$

$\mathscr{D}_4$		$\mathscr{D}_5$							
1	2341	1	**23451**	12	*35412*	23	**25413**	34	*23154*
2	3421	2	**34251**	13	*45132*	24	**54213**	35	*31254*
3	4321	3	**43251**	14	**51432**	25	**45213**	36	**21534**
4	3412	4	**34521**	15	**41532**	26	**54123**	37	**51234**
5	3142	5	**35421**	16	**54132**	27	**51423**	38	**25134**
6	4312	6	**43521**	17	**43152**	28	**45123**	39	**53124**
7	2413	7	**24531**	18	**31452**	29	**24153**	40	**31524**
8	4123	8	**45231**	19	**34152**	30	**41253**	41	**35124**
9	2143	9	**25431**	20	**43512**	31	**21453**	42	**53214**
		10	*54231*	21	**34512**	32	*41523*	43	**35214**
		11	*53421*	22	**53412**	33	*24513*	44	**23514**

In $\mathscr{D}_5$ the sublists $\phi(i,\mathscr{D}_4)$ and $\psi(i,\mathscr{D}_3)$, in direct or reverse order, for some i, $1 \leqslant i \leqslant 4$, are in bold-face and italic, respectively.

or, conversely, by reading this relation left-to-right and replacing i by $i-1$,

$$\psi(i-1,f_{n-2})=\begin{cases}\psi(i,f_{n-2})\cdot\langle i-2,n\rangle\cdot\langle i-1,i\rangle & \text{if } 2<i\leqslant n-1,\\ \psi(i,f_{n-2})\cdot\langle 1,2\rangle\cdot\langle n-1,n\rangle & \text{if } i=2.\end{cases}\tag{4}$$

(ii) *If $n\geqslant 3$ then*

$$\phi(i+1,f_{n-1})=\begin{cases}\phi(i,f_{n-1})\cdot\langle n,i,i-1\rangle & \text{if } 1<i\leqslant n-2,\\ \phi(i,f_{n-1})\cdot\langle 1,n-1,n\rangle & \text{if } i=1\end{cases}\tag{5}$$

or, conversely and by replacing i by $i-1$,

$$\phi(i-1,f_{n-1})=\begin{cases}\phi(i,f_{n-1})\cdot\langle i-2,i-1,n\rangle & \text{if } 2<i\leqslant n-1,\\ \phi(i,f_{n-1})\cdot\langle n,n-1,1\rangle & \text{if } i=2.\end{cases}\tag{6}$$

(iii) *If $n=4$*

$$\psi(i,\ell_2)=\begin{cases}\phi(i,\ell_3)\cdot\langle i,i+1\rangle & \text{if } i\neq 3,\\ \phi(i,\ell_3)\cdot\langle 1,3\rangle & \text{if } i=3.\end{cases}\tag{7}$$

If $n\geqslant 5$

$$\psi(i,\ell_{n-2})=\begin{cases}\phi(i,\ell_{n-1})\cdot\langle n-2,n-3,1\rangle & \text{if } i=1,\ n \text{ even},\\ \phi(i,\ell_{n-1})\cdot\langle 1,n-3,n-2\rangle & \text{if } i=1,\ n \text{ odd},\\ \phi(i,\ell_{n-1})\cdot\langle i-1,i\rangle\cdot\langle n-3,n-2\rangle & \text{if } 2\leqslant i\leqslant n-4,\\ \phi(i,\ell_{n-1})\cdot\langle n-2,n-3,n-4\rangle & \text{if } i=n-3,\\ \phi(i,\ell_{n-1})\cdot\langle n-4,n-3,n-1\rangle & \text{if } i=n-2,\\ \phi(i,\ell_{n-1})\cdot\langle n-4,n-2\rangle\cdot\langle n-3,n-1\rangle & \text{if } i=n-1,\ n \text{ even},\\ \phi(i,\ell_{n-1})\cdot\langle n-1,n-2,n-4\rangle & \text{if } i=n-1,\ n \text{ odd},\end{cases}\tag{8}$$

or conversely

$$\phi(i,\ell_{n-1})=\psi(i,\ell_{n-2})\cdot\pi^{-1}\tag{9}$$

with π the permutation which occurs in the corresponding case in relation (7) or (8). So, for example, if $n\geqslant 5$, $i=n-1$ then:

- $\phi(i,\ell_{n-1})=\psi(i,\ell_{n-2})\cdot\langle n-4,n-2\rangle\cdot\langle n-3,n-1\rangle$ *if n is even (in this case $\pi=\pi^{-1}$),*
- $\phi(i,\ell_{n-1})=\psi(i,\ell_{n-2})\cdot\langle n-1,n-4,n-2\rangle$ *if n is odd.*

Proof. The proof is direct, and consists essentially of checking each case. For brevity, we do not give the proof of (iii) which is similar to the first two cases.

(i) If $1 < i \leqslant n-2$ then

$$\psi(i+1, f_{n-2}) = (2, \ldots, i-1,\ \mathbf{i},\ \mathbf{i+2},\ \mathbf{n},\ i+3, \ldots, 1,\ \mathbf{i+1}),$$

$$\psi(i, f_{n-2}) = (2, \ldots, i-1,\ \mathbf{i+1},\ \mathbf{n},\ \mathbf{i+2},\ i+3, \ldots, 1,\ \mathbf{i})$$

and if $i = 1$ then

$$\psi(2, f_{n-2}) = (\mathbf{3}, \mathbf{n}, 4, \ldots, n-1, \mathbf{1}, \mathbf{2}),$$

$$\psi(1, f_{n-2}) = (\mathbf{n}, \mathbf{3}, 4, \ldots, n-1, \mathbf{2}, \mathbf{1}),$$

where the little numbers are indices of array elements.

(ii) If $1 < i \leqslant n-2$ then

$$\phi(i+1, f_{n-1}) = (2, \ldots, i-1,\ \mathbf{i},\ \mathbf{n},\ i+2, \ldots, 1,\ \mathbf{i+1}),$$

$$\phi(i, f_{n-1}) = (2, \ldots, i-1,\ \mathbf{n},\ \mathbf{i+1},\ i+2, \ldots, 1,\ \mathbf{i})$$

and if $i = 1$ then,

$$\phi(2, f_{n-1}) = (\mathbf{n}, 3, \ldots, n-1, \mathbf{1}, \mathbf{2}),$$

$$\phi(1, f_{n-1}) = (\mathbf{2}, 3, \ldots, n-1, \mathbf{n}, \mathbf{1}). \qquad \square$$

Corollary 5. *Successive derangements in $\mathscr{D}_n$ differ at most in four positions.*

Note that $\mathscr{D}_n$ is a cyclic Gray code.

3. Generating algorithm

The definition given by (2) says that $\mathscr{D}_n$ is the concatenation of many lists, which are all similar in some sense to $\mathscr{D}_{n-1}$ or $\mathscr{D}_{n-2}$. This result is formalized in Lemma 9 below, and our generating algorithm for $\mathscr{D}_n$ is based on it. Now we give some technical definitions.

Two lists are *isomorphic* if, in the first list, a sequence is transformed into its successor via the same permutation as the corresponding sequence in the second list is transformed into its successor; and two lists are *similar* if after erasing the constant entries in the first list, and possibly reversing it, the lists become isomorphic. More formally.

Definition 6. Let $\mathscr{L}$ and $\mathscr{S}$, respectively, be a list of length-n integer sequences and a list of permutations in S_n. We say that $\mathscr{L}$ is isomorphic to $\mathscr{S}$ if:

(1) the lists contain the same number of sequences, say p,

Table 2
$\mathscr{A}$ is isomorphic to $\mathscr{D}_4$ and $\mathscr{B} = \phi(2, \overline{\mathscr{D}_4})$ is $\{1,2,3,4\}$-similar to $\mathscr{D}_4$

$\mathscr{D}_4$	$\mathscr{A}$	$\mathscr{B}$
2341	5341	51432
3421	3451	41532
4321	4351	54132
3412	3415	43152
3142	3145	31452
4312	4315	34152
2413	5413	43512
4123	4153	34512
2143	5143	53412

Note that $\mathscr{A}$ is the reverse of the list obtained by erasing the last entry of each sequence in $\mathscr{B}$.

(2) for all $1 \leqslant j < p$, if σ (resp. τ) is the jth sequence in $\mathscr{L}$ (resp. permutation in $\mathscr{S}$) and σ' (resp. τ') is its successor in $\mathscr{L}$ (resp. $\mathscr{S}$) then $\sigma' = \sigma \cdot \pi$, where π is such that $\tau' = \tau \cdot \pi$.

Definition 7. For a length-n integer sequence list $\mathscr{L}$, a set $T \subseteq [n]$ with $card(T) = m \leqslant n$, and a length-m permutation list $\mathscr{S}$, we say that $\mathscr{L}$ is T-similar to $\mathscr{S}$ if:

(1) for all $i \in [n] \setminus T$, the entry in position i has constant value throughout the list $\mathscr{L}$,
(2) after erasing all entries in positions $i \in [n] \setminus T$ in each sequence in $\mathscr{L}$ the obtained list is isomorphic to $\mathscr{S}$ or to $\overline{\mathscr{S}}$.

In the list $\mathscr{L}$, indices in T and their corresponding entries are called *active* (relative to $\mathscr{S}$).

Clearly, if $\mathscr{L}$ is isomorphic to $\mathscr{S}$ then $\mathscr{L}$ is $[n]$-similar to $\mathscr{S}$. See Table 2 for an example of isomorphic and similar lists.

Lemma 8. *If* $n \geqslant 2$ *then*

(1) *for* $1 \leqslant i \leqslant n$ *the list* $\phi(i, \mathscr{D}_n)$ *is* $\{1, 2, \ldots, n\}$*-similar to* $\mathscr{D}_n$,
(2) *for* $1 \leqslant i \leqslant n+1$ *the list* $\psi(i, \mathscr{D}_n)$ *is* $\{1, 2, \ldots, i-1, i+1, \ldots, n+1\}$*-similar to* $\mathscr{D}_n$.

Proof. Clearly, each derangement in $\phi(i, \mathscr{D}_n)$ has its last position, the $(n+1)$th one, equal to i. Consider σ in $\phi(i, \mathscr{D}_n)$ and τ in $\mathscr{D}_n$ with $\sigma = \phi(i, \tau)$. Then $\sigma' = \phi(i, \tau')$ with σ' and τ' the successor of σ and τ, respectively, and the result holds by considering the form of τ and τ' given in point (1) of Remark 2. (2) This proposition is proved similarly, by considering the point (2) of Remark 2. $\square$

For $U \subset T \subseteq [n]$ we say that U is a *child-subset* (or *c-subset*) of T if: (1) the largest element of T is not in U, and (2) $1 \leqslant card(T \setminus U) \leqslant 2$.

```
procedure gen_up(n,t,r)
var i,run;
begin
tail := pred[tail];
if t = ψ then remove(r); endif
if n = 3
then d := d · ⟨head, succ[head], tail⟩;
else run := head
    for i := 1 to n − 1 do
        if i is odd
        then gen_up(n − 1, φ, run);      // produce a φ(i, D_{n−1})-like list
             update d as in (7) or (8);
             if n > 4 then
             gen_down(n − 2, ψ, run);    // produce a ψ(i, D̄_{n−2})-like list
             endif
             if i ≠ n − 1
             then update d as in (3); run := succ[run];
             endif
        else if n > 4 then
             gen_up(n − 2, ψ, run);      // produce a ψ(i, D_{n−2})-like list
             endif
             update d as in (9);
             gen_down(n − 1, φ, run);    // produce a φ(i, D̄_{n−1})-like list
             if i ≠ n − 1
             then update d as in (5); run := succ[run];
             endif
        endif
    enddo
endif
if t = ψ then append(r); endif
tail := succ[tail];
end
```

Fig. 2. Generating derangements in Gray code order.

Lemma 9. *Let $\mathscr{L}$ be a length-n sequence list, $T \subseteq [n]$, and $m = card(T) \geqslant 4$. If $\mathscr{L}$ is T-similar to the derangement list $\mathscr{D}_m$ then $\mathscr{L} = \mathscr{L}_1 \circ \mathscr{L}_2 \circ \cdots \circ \mathscr{L}_{2(m-1)}$ where each sublist $\mathscr{L}_j$ is U_j-similar to the derangement list $\mathscr{D}_{card(U_j)}$, with U_j a c-subset of T.*

Proof. By Lemma 8 above and applying recursively relation (2). □

Obviously, $\mathscr{D}_n$ is $\{1,2,\dots,n\}$-similar to itself and the procedure *gen_up* in Fig. 2 generates the list $\mathscr{D}_n$ according to the lemma above: the lists $\mathscr{L}_j$ are produced iteratively, and each of them is generated recursively. So, each call of this procedure fills up entries with indices in an active set $T \subseteq [n]$ associated with it, and in a recursive call T is replaced by a c-subset of T.

In our algorithm, the set T of active indices is represented by four global variables: the integers *head* and *tail* and the arrays *succ* and *pred*, defined as follows. If at a computational step $T = \{i_1, i_2, \dots, i_k\}$, then $head = i_1$, $tail = i_k$, $succ[i_j] = i_{j+1}$ and $pred[i_j] = i_{j-1}$.

If the active set associated with the current call is T, then the call of $gen\_up(j,t,run)$ initiated by the current call generates a sublist that is U-similar to $\mathscr{D}_j$, where

$$U = \begin{cases} T \setminus \{tail\} & \text{if } t = \phi, \\ T \setminus \{run, tail\} & \text{if } t = \psi \end{cases}$$

(recall that $tail = \max(T)$).

Less formally, $gen\_up(j,\phi,run)$ produces a '$\phi(i,\mathscr{D}_j)$-*like*' list, and $gen\_up(j,\psi,run)$ produces a '$\psi(i,\mathscr{D}_j)$-*like*' list (see relation (2)), where *run* is the *i*th element in the set T. The call of *gen_down* works as *gen_up* except $\mathscr{D}_j$ is replaced by $\overline{\mathscr{D}_j}$.

For a simpler expression of the generating algorithm we consider initially the active set $T=[n+1]$, and each call begins by removing *tail*, the largest element in T. Thus, before the first call of the generating procedure, the variables which correspond to T are: $head = 1$, $tail = n+1$, $succ[i] = i+1$ for $1 \leqslant i \leqslant n$, and $pred[i] = i-1$ for $2 \leqslant i \leqslant n+1$. The current derangement is stored in a global variable d, initialized by $d = first(\mathscr{D}_n)$.

The main call $gen\_up(n,\phi,0)$ produces the list $\mathscr{D}_n$, $n \geqslant 3$, and the value $r=0$ is for convenience; in fact when $t=\phi$ the value of r is not required. The procedure which generates the reverse list $\overline{\mathscr{D}_n}$ is called *gen_down*, shown in the appendix, and essentially executes the statements of *gen_up* in reverse order and replaces the calls of *gen_up* by *gen_down* and vice versa. Procedures $remove(r)$ and $append(r)$, also shown in the appendix, remove and append r in the current active set (given by the variables *head*, *tail*, *succ* and *pred*).

Between any successive calls at least one update statement is performed, and after each update statement (including the case $n=3$) a new derangement is produced and printed out. The current derangement d is transformed into its successor according to relations (3), (5), (7), (8) or (9) in Lemma 4. More precisely, the current derangement is subject to the transformation given in the appropriate case of Lemma 4, and it acts on the active indices.

For example, in our algorithm relation (5) becomes:

if $i = 1$
then $d := d \cdot \langle head, pred[tail], tail\rangle$;
else $d := d \cdot \langle tail, run, pred[run]\rangle$;
endif

Clearly, the time complexity of *gen_up* is proportional to the total number of recursive calls. Since each call produces at least one new derangement the time complexity of $gen\_up(n,t,r)$ is in $\mathcal{O}(d_n)$. A C implementation of our algorithm is available at `http://www.u-bourgogne.fr/v.vincent/AA/`.

4. Permutations with a given number of fixed points

Here, we generalize the Gray code in the previous sections to permutations with a given number of fixed points and permutations with a bounded number of fixed points.

Let $c=(c(1),c(2),\dots,c(n))$ be an n-combination of m, $n\leqslant m$, in integer sequence representation, so that $1\leqslant c(i)<c(i+1)\leqslant m$ for $i=1,2,\dots,n-1$. Also, let $t_c=t$ be the binary representation of c, i.e., $t=(t(1),t(2),\dots,t(m))$ with $t(i)=1$ if there exists a j such that $c(j)=i$, and $t(i)=0$ elsewhere. With those notations, for a derangement $d=(d(1),d(2),\dots,d(n))\in D_n$ we define the length-m sequence $u=(u(1),u(2),\dots,u(m))$, denoted by $\sqcup(c;d)$, as

$$u(i)=\begin{cases} i & \text{if } t(i)=0,\\ c(d(j)) & \text{if } t(i) \text{ is the } j\text{th } 1 \text{ in } t \end{cases} \tag{10}$$

and we call $u=\sqcup(c;d)$ the *shuffle* of c by d on the trajectory t.

In other words, u acts on indices $c(1),c(2),\dots,c(n)$ as d, and fixes the other indices. The shuffle operator over combinatorial objects was formally defined in a larger context in [15,16]. It is not hard to show that $\sqcup(c;d)$ is a permutation of $[m]$ with exactly n "deranged" points (i.e. with exactly $m-n$ fixed points), and in addition, each such permutation can be uniquely constructed by shuffle operation from an appropriate combination and a derangement. More formally, if $u=(u(1),u(2),\dots,u(m))$ is a permutation of $[m]$ with exactly $m-n$ fixed points then $u=\sqcup(c;d)$, where

- $c=(c(1),c(2),\dots,c(n))$ is the n-combination of m corresponding to the subset of $[m]$ where $u(i)\neq i$, and
- d is the normal form of the sequence $(u(c(1)),u(c(2)),\dots,u(c(n)))$.

Example. If $n=3$, $m=6$, $c=(2,5,6)$, and $d=(2,3,1)$ then $t=(0,\mathbf{1},0,0,\mathbf{1},\mathbf{1})$ and $\sqcup(c;d)=(1,\mathbf{5},3,4,\mathbf{6},\mathbf{2})$; or if $n=4$, $m=6$, $c=(1,2,4,5)$, and $d=(2,4,1,3)$ then $t=(\mathbf{1},\mathbf{1},0,\mathbf{1},\mathbf{1},0)$ and $\sqcup(c;d)=(\mathbf{2},\mathbf{5},3,\mathbf{1},\mathbf{4},6)$.

See also [18] for a similar approach. To summarize, we have:

Lemma 10. *If $C_{m,n}$ is the set of all n-combinations of $[m]$ and $S_{m,n}$ the set of all permutations of $[m]$ with exactly $m-n$ fixed points then*

$$\sqcup : C_{m,n}\times D_n \to S_{m,n}$$

defined by (10) *is a bijection.*

Also, we extend the shuffle operation in a natural way to lists of derangements: if $\mathscr{D}=d_1,d_2,\dots$ is a sublist of $\mathscr{D}_n$ and $c\in C_{m,n}$ then $\sqcup(c;\mathscr{D})$ is the list $\sqcup(c;d_1),\sqcup(c;d_2),\dots,$ and $\sqcup(c;\overline{\mathscr{D}})=\overline{\sqcup(c;\mathscr{D})}$.

A *strong* Gray code for the set $C_{m,n}$ of n-combinations of m, in integer sequence representation, is a list for $C_{m,n}$ where two successive sequences, say $c=(c(1),c(2),\dots,c(n))$ and $c'=(c'(1),c'(2),\dots,c'(n))$, are such that, for some $1\leqslant j\leqslant m$, $c(i)=c'(i)$ for all $i\neq j$; see [3,5,12] for such a Gray code.

Lemma 11. *If $\mathscr{C}_{m,n}$ is a strong Gray code for the set $C_{m,n}$ then the list $\mathscr{S}_{m,n}$ defined by*

$$\mathscr{S}_{m,n}=\bigcirc_{c\ \text{in}\ \mathscr{C}_{m,n}} \sqcup(c;\mathscr{D}_n^{(r)}) \tag{11}$$

is a Gray code for the set $S_{m,n}$, where r is the rank of c in $\mathscr{C}_{m,n}$ (the first combination in $\mathscr{C}_{m,n}$ has rank zero) and $\mathscr{D}_n^{(r)}$ is $\mathscr{D}_n$ or $\overline{\mathscr{D}}_n$ according as r is even or odd.

Proof. The list $\mathscr{S}_{m,n}$ has no repetitions, and, disregarding the order, it equals the set $S_{m,n}$. Moreover, for a fixed c in $C_{m,n}$, the Hamming distance between two derangements in D_n, say d and d', equals the Hamming distance between $\sqcup(c;d)$ and $\sqcup(c;d')$. So, any successive permutations in $\sqcup(c;\mathscr{D}_n)$—or equivalently in $\sqcup(c;\overline{\mathscr{D}}_n)$—differ in at most four positions. If c' is the successor of c in $\mathscr{C}_{m,n}$ then $t' = t_{c'}$ and $t = t_c$, the binary representations of c' and c, differ in exactly two positions, say k and ℓ, with $t(k)=t'(\ell)=0$ and $t'(k)=t(\ell)=1$. Since $\mathscr{C}_{m,n}$ is a strong Gray code, the permutations $\sigma=\sqcup(c;d)$ and $\sigma=\sqcup(c';d)$ differ in exactly three positions, namely k,ℓ and i, where i is such that $\sigma(i)=\ell$ and $\sigma'(i)=k$. Moreover, the index i can be computed in constant time if d is the first or last derangement in $\mathscr{D}_n$. □

The next lemma extends the result of the previous one to permutations where the number of fixed points is bounded between two constants. In this case $\mathscr{C}_{m,n}$ denotes the Eades–McKay Gray code for combinations, and it has (see [5,12])

- $\mathit{first}(\mathscr{C}_{m,n}) = (1,2,\ldots,n)$, and
- $\mathit{last}(\mathscr{C}_{m,n}) = (m-n+1, m-n+2,\ldots,m)$.

Lemma 12. *Let $1 \leqslant k \leqslant \ell \leqslant m$ and $S_{m,k,\ell}$ be the set of all permutations in S_m with i "deranged" points, $k \leqslant i \leqslant \ell$. Then the list*

$$\mathscr{S}_{m,k,\ell} = \bigcirc_{i=k}^{\ell} \mathscr{S}_{m,i}^{(k-i)} \tag{12}$$

is a Gray code for the set $S_{m,k,\ell}$.

Proof. It is sufficient to prove that the last permutation in $\mathscr{S}_{m,i}^{(k-i)}$ and the first one in $\mathscr{S}_{m,i+1}^{(k-i+1)}$ differ in at most four positions. But $\mathit{last}(\mathscr{S}_{m,i}^{(k-i)})=\sqcup(c,e_i)$, and $\mathit{first}(\mathscr{S}_{m,i+1}^{(k-i+1)}) = \sqcup(c',e_{i+1})$, with

(i) $c = \mathit{last}(\mathscr{C}_{m,i})$ and $c' = \mathit{last}(\mathscr{C}_{m,i+1})$ if $k-i$ is even, or
(ii) $c = \mathit{first}(\mathscr{C}_{m,i})$ and $c' = \mathit{first}(\mathscr{C}_{m,i+1})$ if $k-i$ is odd,

and $e_j = f_j$ or $e_j = \ell_j$; see Lemma 3 and the remark that follows. If $u=\sqcup(c,e_i)$ and $u' = \sqcup(c',e_{i+1})$ then: in case (i), $u(j)=u'(j)=j$ for all $j=1,2,\ldots,m-i-1$ and $u(j)=u'(j)=j+1$ for all $j=m-i+1, m-i+2,\ldots,m-3$; in case (ii), $u(j)=u'(j)=j+1$ for all $j=1,2,\ldots,i-3$ and $u(j)=u'(j)=j$ for all $j=i+2,i+3,\ldots,m$. In both cases u differs from u' in at most four positions. □

Algorithmic considerations

The lists $\sqcup(c;\mathscr{D}_n^{(r)})$ in (11) is c-similar to $\mathscr{D}_n$ (c is regarded as a subset of $[m]$) and the procedure *gen_up* and *gen_down* can easily be transformed to generate these lists. In addition, with an efficient algorithm to compute the successor of c in $\mathscr{C}_{m,n}$ and with appropriate initial values for the variables and transition statements between lists, the iterative call of *gen_up* and *gen_down* produces $\mathscr{S}_{m,n}$ in constant average time. See [16,18] for loopless generating algorithms for $\mathscr{C}_{m,n}$. Similar considerations hold for the list $\mathscr{S}_{m,k,\ell}$ defined in relation (12). The loopless generation of those lists remains an open problem.

Acknowledgements

The authors thank the referees for various comments whose suggestions greatly improved the present paper.

Note added in proof. We recently learned of Korsh' and LaFollette's [10] algorithm for generating derangements. Their algorithm has the remarkable properties (a) that successive permutations differ by only one transposition or one rotation of three elements and (b) it is loopless. Our algorithm is based on a recursive counting relation and so has the advantage of being simpler to describe.

Appendix.

```
procedure gen_down(n, t, r)
var i, run;
begin
tail := pred[tail];
if t = ψ then remove(r); endif
if n = 3 then d := d · ⟨head, tail, succ[head]⟩;
else run := pred[tail]
  for i := n − 1 downto 1 do
    if i is odd
    then if n > 4 then
      gen_up(n − 2, ψ, run);
      endif
      update d as in (9);
      gen_down(n − 1, φ, run);
      if i ≠ 1
      then update d as in (6); run := pred[run];
      endif
    else gen_up(n − 1, φ, run);
      update d as in (7) or (8);
      if n > 4 then
      gen_down(n − 2, ψ, run);
      endif
      update d as in (4);
      run := pred[run];
```

```
    endif
  enddo
endif
if t = ψ then append(r); endif
tail := succ[tail];
end

procedure remove(r)
begin
if r = head
then head := succ[r];
else if r = tail
  then tail := pred[tail]
  else succ[pred[r]] := succ[r];
    pred[succ[r]] := pred[r];
  endif
endif
end

procedure append(r)
begin
if r < head
then head := pred[head];
else if r > tail
  then tail := succ[tail]
  else succ[pred[r]] := r;
    pred[succ[r]] := r;
  endif
endif
end
```

References

[1] S.G. Akl, A new algorithm for generating derangements, BIT 20 (1980) 2–7.

[2] S.G. Akl, The Design and Analysis of Parallel Algorithms, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[3] P.J. Chase, Combination generation and graylex ordering, Congr. Numer. 69 (1989) 215–242.

[4] L. Comtet, Advanced Combinatorics: The Art of Finite and Infinite Expansions, Reidel, Dordrecht, Holland, 1974.

[5] P. Eades, B. McKay, An algorithm for generating subsets of fixed size with a strong minimal change property, IPL 19 (1984) 131–133.

[6] S. Effler, F. Ruskey, A CAT algorithm for generating permutations with a fixed number of inversions, Inform. Process. Lett. 86 (2) (2003) 107–112.

[7] D.E. Knuth, The Art of Computer Programming, Combinatorial Algorithms, Vol. 4. Pre-fascicle 2b (Generating all permutations), http://www-cs-staff.Stanford.EDU/knuth/taocp.html#vol4,2002.

[8] C.W. Ko, F. Ruskey, Generating permutations of a bag by interchanges, IPL 41 (5) (1992) 263–269.

[9] J.F. Korsh, Loopless generation of up-down permutations, Discrete Math. 240 (1–3) (2001) 97–122.

[10] J. Korsh, P.S. LaFollette, Constant time generation of derangements, Department Technical Report, Temple University CIS, December 2002.
[11] D. Roelants van Baronaigien, F. Ruskey, Generating permutations with given ups and downs, Discrete Appl. Math. 36 (1) (1992) 57–65.
[12] F. Ruskey, Simple combinatorial Gray codes constructed by reversing sublists in ISAAC Conference, Lecture Notes in Computer Science, Vol. 762, Springer, Berlin, 1993, pp. 201–208.
[13] R. Sedgewick, Permutation generation methods, Comput. Surveys 9 (2) (1977) 137–164.
[14] R. Stanley, Enumerative Combinatorics, Vol. 1, Cambridge University Press, Cambridge, England, 1997.
[15] V. Vajnovszki, Gray visiting Motzkins, Acta Inform. 38 (2002) 793–811.
[16] V. Vajnovszki, A loopless algorithm for generating the permutations of a multiset, TCS, 307 (2003) 415–431.
[17] T. Walsh, Loop-free sequencing of bounded integer compositions, J. Combin. Math. Combin. Comput. 33 (2000) 323–345.
[18] T. Walsh, Gray codes for involutions, J. Combin. Math. Combin. Comput. 36 (2001) 95–118.