Electronic Journal of Statistics
Vol. 8 (2014) 883–893
ISSN: 1935-7524
DOI: 10.1214/14-EJS913

Random Latin squares and Sudoku designs generation

Roberto Fontana

Dept. of Mathematical Sciences
Politecnico di Torino – Italy
e-mail: roberto.fontana@polito.it

Abstract: Uniform random generation of Latin squares is a classical problem. In this paper we prove that both Latin squares and Sudoku designs are maximum cliques of properly defined graphs. We have developed a simple algorithm for uniform random sampling of Latin squares and Sudoku designs. The corresponding SAS code is available in the supplementary material.

AMS 2000 subject classifications: Primary 62K15.
Keywords and phrases: Latin squares, Sudoku designs, random generation, cliques.

Received May 2013.

1. Introduction

Generating uniformly distributed random Latin squares is a a topic with a long history in the design of experiments and related fields. Already in 1933, F. Yates [10] wrote

> ...it would seem theoretically preferable to choose a square at random from all the possible squares of given size.

The widely used algorithm for generating random Latin squares of a given order is given in [7]. It is based on a proper set of *moves* that connect all the squares and make the distribution of visited squares *approximately* uniform.

In this paper, we present a new approach that is based on the equivalence between Latin squares and maximum cliques of a graph. This approach is also valid for Sudoku designs.

The paper is organized as follows. In Section 2, the equivalence between Latin squares (Sudoku designs) and maximum cliques of a suitable graph is demonstrated. Section 3 describes an algorithm for generating uniformly distributed random Latin squares and Sudoku designs. The corresponding SAS code is available in the supplementary material [6]. Concluding remarks are made in Section 4.

2. Latin squares and Sudoku designs are maximum cliques

2.1. Latin squares

A Latin square of order n is an $n \times n$ matrix L_n in which each of n distinct symbols appear n times, once in each row and one in each column. For the sake of simplicity we consider the integers $1, 2, \ldots, n$ as symbols. We denote by $L_n[\cdot, c], c = 1, \ldots, n$ ($L_n[r, \cdot], r = 1, \ldots, n$) the columns (the rows) of L_n and by $\mathcal{L}_n$ the set of all the Latin squares of order n.

For example, a Latin square of order 4, $L_4 \in \mathcal{L}_4$, is given by

$$L_4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{pmatrix}. \tag{1}$$

We can look at a Latin square $L_n = [\ell_{rc}; r, c = 1, \ldots, n]$ as a set of n disjoint permutation matrices, following the approach recently adopted in [4] and [5]. For each symbol s, $s = 1, \ldots, n$ we consider the $n \times n$ matrix $P^{(s)} = [p_{rc}^{(s)}; r, c = 1, \ldots, n]$ where

$$p_{rc}^{(s)} = \begin{cases} 1 & \text{if } \ell_{rc} = s \\ 0 & \text{otherwise} \end{cases}.$$

Given a permutation matrix P the corresponding permutation $\pi = (\pi_1, \ldots, \pi_n)$ of $(1, \ldots, n)$ is defined as

$$\pi = P 1_n$$

where 1_n is the $n \times 1$ column vector whose elements are $1, \ldots, n$. Viceversa, given a permutation $\pi = (\pi_1, \ldots, \pi_n)$ of $(1, \ldots, n)$ the corresponding permutation matrix $P = [p_{rc}; r, c = 1, \ldots, n]$ is defined as

$$p_{rc} = \begin{cases} 1 & \text{if } c = \pi_r \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

We denote by ϕ the function that transforms a permutation π of $(1, \ldots, n)$ into a permutation matrix $P = \phi(\pi)$ according to Equation (2).

For the Latin square $L_4 \in \mathcal{L}_4$ of Equation (1), the permutation matrix $P^{(2)}$ is given by

$$P^{(2)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

and the corresponding permutation $\pi^{(2)}$ of $(1, 2, 3, 4)$ is

$$\pi^{(2)} = (2, 3, 4, 1).$$

It immediately follows that a Latin square of order n can be written as

$$L_n = P^{(1)} + 2P^{(2)} + \cdots + nP^{(n)} \tag{3}$$

where $P^{(s)}, s = 1, \ldots, n$ are *mutually disjoint* permutation matrices. Two permutation matrices $P^{(s)}$ and $P^{(t)}$ are disjoint if and only if $p_{rc}^{(s)} p_{rc}^{(t)} = 0$ for each $r, c \in \{1, \ldots, n\}$. Equivalently two permutations $\pi^{(s)}$ and $\pi^{(t)}$ are disjoint if and only if $\pi_r^{(s)} \neq \pi_r^{(t)}$ for $r = 1, \ldots, n$.

Without loss of generality, as we will explain below, assume that $P^{(1)} = I_n$ where I_n is the $n \times n$ identity matrix. The permutation $\pi^{(1)}$ corresponding to $P^{(1)}$ is the identity permutation ι_n, $\pi^{(1)} \equiv \iota_n = (1, \ldots, n)$.

Let us denote by $\mathcal{P}_n$ the set of all the permutations of $\{1, \ldots, n\}$ and, given $\pi \in \mathcal{P}_n$, by $\mathcal{L}_n^\pi \subset \mathcal{L}_n$ the set of all the Latin squares of order n for which $P^{(1)} = \phi(\pi)$ and $P^{(2)} < \cdots < P^{(n)}$ where $P^{(s)} < P^{(t)}$ or, equivalently, $\pi^{(s)} < \pi^{(t)}$ means that $(\pi_1^{(s)}, \ldots, \pi_n^{(s)}) <_{lex} (\pi_1^{(t)}, \ldots, \pi_n^{(t)})$. The symbol "$<_{lex}$" denotes the standard lexicographic order, $(a_1, \ldots, a_n) <_{lex} (b_1, \ldots, b_n) \Leftrightarrow \exists m > 0 \ \forall i < m \ \ a_i = b_i$ and $a_m < b_m$. For simplicity we will write "$<$" in place of "$<_{lex}$". As it will become clear later on, any order between permutations can be chosen.

Let us consider $\mathcal{L}_n^{\iota_n}$, the set of all the Latin squares of order n for which $P^{(1)} = I_n$ and $P^{(2)} < \cdots < P^{(n)}$. As $\mathcal{L}_n^{\iota_n}$ is built, we can generate all the Latin squares of order n, $L_n \in \mathcal{L}_n$, considering

1. all the $(n-1)!$ permutations $(s_2, \ldots, s_n)$ of the symbols $2, \ldots, n$ and assigning them to the permutation matrices $P^{(2)}, \ldots P^{(n)}$
$$I_n + s_2 P^{(2)} + \cdots + s_n P^{(n)};$$
2. all the $n!$ sets $\mathcal{L}_n^\pi$ where $\pi \in \mathcal{P}_n$ is a permutation of $(1, \ldots, n)$. We observe that $\mathcal{L}_n^\pi$ contains all the Latin squares that are generated permuting the columns of a Latin square L_n of $\mathcal{L}_n^{\iota_n}$
$$\mathcal{L}_n^\pi = \{[L_n[\cdot, \pi_1] | \ldots | L_n[\cdot, \pi_n]] : L_n \in \mathcal{L}_n^{\iota_n}\}.$$

It follows that in order to generate a random Latin square L_n it is sufficient:

1. to generate a random Latin square $L_n^{(1)} \in \mathcal{L}_n^{\iota_n}$;
2. to generate $L_n^{(2)}$ by a random permutation of the symbols $2, \ldots, n$ of $L_n^{(1)}$;
3. to generate L_n by a random permutation of the columns of $L_n^{(1)}$.

We observe that the number $\#\mathcal{L}_n$ of Latin squares of order n is
$$\#\mathcal{L}_n = n!(n-1)!\#\mathcal{L}_n^{\iota_n}. \tag{4}$$
It is worth noting that the suggested approach, making use of symbols and columns permutations, allows us to reduce the number of Latin squares to be explicitly considered from $\#\mathcal{L}_n$ to $\#\mathcal{L}_n^{\iota_n}$ with a reduction factor equal to $n!(n-1)!$. More specifically, the third step of the procedure above is based on the assigment of the initial permutation matrix $P^{(1)} = I_n$.

To generate a Latin square $L_n \in \mathcal{L}_n^{\iota_n}$ we have to build $n-1$ permutation matrices $P^{(s)}, s = 2, \ldots, n$, $P^{(2)} < \cdots < P^{(n)}$, that are mutually disjoint and that are disjoint with I_n. In the language of permutations, a permutation δ that is disjoint with the identity permutation ι_n, i.e. $\delta_r \neq r, r = 1, \ldots, n$, is called a *derangement*. The permutation matrix $P = [p_{rc}; r, c = 1, \ldots, n]$ of a derangement δ, $P = \phi(\delta)$, satisfies the condition $p_{rr} = 0, r = 1, \ldots, n$.

It follows that we have to build $n-1$ derangements $\delta^{(s)}, s = 2, \ldots, n$ of $(1, \ldots, n)$, $\delta^{(2)} < \cdots < \delta^{(n)}$ such that $\delta_r^{(s)} \neq \delta_r^{(t)}, r = 1, \ldots, n$ for each $s, t \in \{2, \ldots, n\}, s \neq t$.

Let $\mathcal{D}_n \subset \mathcal{P}_n$ be the set of all the derangements of $(1,\ldots,n)$: we denote by d_n the number of derangements of $(1,\ldots,n)$, $d_n = \#\mathcal{D}_n$.

Let $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$ be the undirected graph whose set of vertices $\mathcal{V}_n$ is the set of derangements $\mathcal{D}_n$ and whose set of edges $\mathcal{E}_n$ contains all the couples of derangements $(\delta^{(i)}, \delta^{(j)}), i < j$ such that $\delta_r^{(i)} \neq \delta_r^{(j)}, r = 1,\ldots,n$. The following theorem, which to the best of our knowledge is new, holds.

Theorem 1. *The Latin squares L_n of order n of $\mathcal{L}_n^{\iota_n}$ are the ordered cliques C_{n-1} of size $n-1$ of $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$*

$$C_{n-1} = (\delta^{(2)}, \ldots, \delta^{(n)}), \;\; \delta^{(2)} < \cdots < \delta^{(n)}.$$

C_{n-1} are the largest cliques of $\mathcal{G}_n$.

Proof. A Latin square $L_n \in \mathcal{L}_n^{\iota_n}$ can be written as

$$L_n = I_n + 2P^{(2)} + \cdots + nP^{(n)}$$

where $P^{(s)}$ is the permutation matrix corresponding to the derangement $\delta^{(s)} = P^{(s)}1_n, s = 2,\ldots,n$ and $\delta^{(2)} < \cdots < \delta^{(n)}$. The derangements $\delta^{(s)}$ are disjoint. It follows that $\{\delta^{(2)},\ldots,\delta^{(n)}\}$ is a clique of $\mathcal{G}_n$. Viceversa, given the clique $\{\delta^{(2)},\ldots,\delta^{(n)}\}$ with $\delta^{(2)} < \cdots < \delta^{(n)}$ we build

$$L_n^\star = I_n + 2\phi(\delta^{(2)}) + \cdots + n\phi(\delta^{(n)})$$

where ϕ is defined in Equation (2). It is immediately evident that $L_n^\star = L_n$.

Finally, Latin squares correspond to the largest cliques because it is evident that it is not possible to find a set of $m > n$ derangements of $\{1,\ldots,n\}$ that are disjoint. □

2.2. Sudoku designs

For the definition of Sudoku designs we refer to [1]

> In 1956, W. U. Behrens ([2]) introduced a specialisation of Latin squares which he called *gerechte*. The $n \times n$ grid is partitioned into n regions, each containing n cells of the grid; we are required to place the symbols $1,\ldots,n$ into the cells of the grid in such a way that each symbol occurs once in each row, once in each column, and once in each region. The row and column constraints say that the solution is a Latin square, and the last constraint restricts the possible Latin squares. By this point, many readers will recognize that solutions to Sudoku puzzles are examples of *gerechte* designs, where $n = 9$ and the regions are the 3×3 subsquares. (The Sudoku puzzle was invented, with the name "number place", by Harold Garns in 1979.)

Analogously to Latin squares, we describe a Sudoku design in terms of Sudoku permutation matrices, as discussed in [4] and [5].

Let us define the regions in which the matrix is divided. We will refer to regions as *boxes*. Let us consider a $n \times n$ matrix, where $n = p^2$ and p is a positive integer. Its row and column positions (i, j) are coded with the integer from 0 to p^2-1. We define boxes $B_{k,m},\ k,m=0,\dots,p-1$ as the following sets of positions

$$B_{k,m}=\{(i,j): kp\le i<(k+1)p, mn\le j<(m+1)n\}\,.$$

It follows that any $n\times n$ matrix A can be partitioned into submatrices A_{km} corresponding to boxes $B_{k,m}$.

An $n\times n$ matrix S_n is a Sudoku, if in each row, in each column and in each box, each of the integers $1,\dots,n$ appears exactly once. We denote by $\mathcal{S}_n$ the set of all the $n\times n$ Sudokus. In Sudoku literature, the set of boxes $B_{b,m},\ m=0,\dots,p-1$ constitutes the b^{th} *band*, $b=0,\dots,p-1$, while the set of boxes $B_{k,s},\ k=0,\dots,p-1$ constitutes the s^{th} *stack*, $s=0,\dots,p-1$.

Let us define a Sudoku permutation matrix $\tilde{P}$, referred to as an S-matrix $\tilde{P}$, as a permutation matrix of order n which has exactly one "1" in each submatrix $P_{k,m}$ corresponding to boxes $B_{k,m}$, $k,m=0,\dots,p-1$. Let us denote by $\tilde{\mathcal{P}}_n\subset\mathcal{P}_n$ the set of all Sudoku permutations. An $n\times n$ Sudoku S_n identifies n matrices $\tilde{P}^{(i)},\ i=1,\dots,n$, where $\tilde{P}^{(i)}$ is the S-matrix corresponding to the positions occupied by the integer i. It follows that a Sudoku $S_n\in\mathcal{S}_n$ can be written as

$$S_n=\tilde{P}^{(1)}+2\tilde{P}^{(2)}+\cdots+n\tilde{P}^{(n)}. \tag{5}$$

We observe that $\tilde{P}^{(1)},\dots,\tilde{P}^{(n)}$ are mutually disjoint and that Equation (5) is the analogue of Equation (3) for Sudoku designs.

We observe that the identity permutation ι_n is not a Sudoku permutation, apart from the trivial $n=2$ case.

We can easily generate S-matrices. Let us define a more compact representation of an S-matrix S, by building the $p\times p$ matrix $S^\star$, whose elements are the only possible position, within each box, where S is equal to 1. Among the S-matrices we can define $S^\star_{0,n}$ whose elements $(S^\star_{0,n})_{km},\ k,m=1,\dots,p$ are

$$(S^\star_{0,n})_{km}=(m,k).$$

For the $n=4$ case we obtain

$$S^\star_{0,4}=\begin{bmatrix}(1,1) & (2,1)\\ (1,2) & (2,2)\end{bmatrix}$$

and the corresponding S-matrix $S_{0,4}$ is

$$S_{0,4}=\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}.$$

For the $n=9$ case we obtain

$$S^\star_{0,9}=\begin{bmatrix}(1,1) & (2,1) & (3,1)\\ (1,2) & (2,2) & (3,2)\\ (1,3) & (2,3) & (3,3)\end{bmatrix}$$

and the corresponding S-matrix $S_{0,9}$ is

$$S_{0,9} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

We will denote by $\sigma_{0,n}$ the permutation corresponding to the S-matrix $S_{0,n}$, $\sigma_{0,n} = S_{0,n}1_n$. It will play the role of the identity permutation ι_n for Sudoku designs.

All the S-matrices can be generated by permuting the rows within each band and the columns within each stack. It follows that the total number of S-matrices is $p!^{2p}$, [4].

Let $\tilde{\mathcal{G}}_n = (\tilde{\mathcal{V}}_n, \tilde{\mathcal{E}}_n)$ be the undirected graph whose set of vertices $\tilde{\mathcal{V}}_n$ is the set of derangements of $\sigma_{0,n}$ *that are also S-permutations*, briefly Sudoku-derangements, and whose set of edges $\tilde{\mathcal{E}}_n$ contains all the couple of Sudoku-derangements $(\tilde{\delta}^{(i)}, \tilde{\delta}^{(j)}), i < j$ such that $\tilde{\delta}_r^{(i)} \neq \tilde{\delta}_r^{(j)}, r = 1, \ldots, n$.

Theorem 1 holds if we replace the graph $\mathcal{G}_n$ with the graph $\tilde{\mathcal{G}}_n$. We observe that $\tilde{\mathcal{G}}_n$ is a subgraph of $\mathcal{G}_n$.

Let us denote by $\mathcal{S}_n^{\sigma_{0,n}}$ the set of all the Sudokus of order n for which $\tilde{P}^{(1)} = S_{0,n}$ and $\tilde{P}^{(2)} < \cdots < \tilde{P}^{(n)}$.

As $\mathcal{S}_n^{\sigma_{0,n}}$ is built, we can generate all the Sudokus of order $n = p^2$ considering

1. all the $(n-1)!$ permutations $(s_2, \ldots, s_n)$ of the symbols $2, \ldots, n$ and assigning them to the permutation matrices $\tilde{P}^{(2)}, \ldots \tilde{P}^{(n)}$
$$I_n + s_2\tilde{P}^{(2)} + \cdots + s_n\tilde{P}^{(n)}$$
2. all the $p!^{2p}$ sets $\mathcal{S}_n^{\sigma}$ where σ is a Sudoku permutation of $(1, \ldots, n)$.

For the total number of Sudokus of order n we get Equation (6) which is the equivalent of Equation (4):

$$\#\mathcal{S}_n = (n-1)!p!^{2p}\#\mathcal{S}_n^{\sigma_{0,n}} \tag{6}$$

3. An algorithm for random sampling

3.1. Latin squares

The algorithm takes n as input and gives L_n, a random Latin square of order n, as output.

The main steps of the algorithm are as follows.

1. Build the undirected graph $\mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$:
   (a) generate $\mathcal{V}_n \equiv \mathcal{D}_n$, the set of all the derangements $\delta^{(i)}, i = 1, \ldots, d_n$ of $\{1, \ldots, n\}$;
   (b) generate $\mathcal{E}_n$, the set of all the edges corresponding to all the couples of derangements $(\delta^{(i)}, \delta^{(j)}), i < j$ such that $\delta_r^{(i)} \neq \delta_r^{(j)}, r = 1, \ldots, n$.
2. Generate all the largest cliques of $\mathcal{G}_n$.
3. Randomly extract one of the largest clique and order its vertices lexicographically. Let use denote this ordered clique by $C_{n-1} = (\delta^{(2)}, \ldots, \delta^{(n)})$. The corresponding Latin square is
$$L_n^{(1)} = I_n + 2\phi(\delta^{(2)}) + \cdots + n\phi(\delta^{(n)}).$$
4. Randomly choose one permutation $\sigma = (s_2, \ldots, s_n)$ of $(2, \ldots, n)$ and generate
$$L_n^{(2)} = I_n + s_2\phi(\delta^{(2)}) + \cdots + s_n\phi(\delta^{(n)}).$$
5. Randomly choose one permutation γ of $(1, \ldots, n)$ and generate L_n permuting the columns of $L_n^{(2)}$ according to γ.

It should be pointed out that in the SAS code the generic derangement δ is stored as the coordinate vector $(\delta_1, \ldots, \delta_n)$.

We now describe the algorithm for $n = 5$.

1. We generate $\mathcal{D}_5$ taking all the permutations δ of $(1, \ldots, 5)$ such that $\delta_r \neq r$, $r = 1, \ldots, 5$. $\mathcal{D}_5$ contains 44 derangements. We denote by $\delta^{(i)}, i = 1, \ldots, 44$ the elements of $\mathcal{D}_5$.
2. We generate $\mathcal{E}_5$ considering all the $\binom{44}{2} = 946$ couples of derangements $(\delta^{(s)}, \delta^{(t)})$, $\delta^{(s)} < \delta^{(t)}$ such that $\delta_r^{(s)} \neq \delta_r^{(t)}, r = 1, \ldots, 5$. We find 276 edges. The graph $\mathcal{G}_5$, generated using the function `tkplot` of the R package igraph, [3], is shown in Figure 1.
3. We use the function `largest.cliques` of the R package igraph, [3], to get all the largest cliques of $\mathcal{D}_{0,5}$. Equivalently we can use the Optnet procedure of SAS/OR, [11] or Cliquer, [8]. We find 56 cliques of size 4.
4. We randomly choose one clique C_4 and we order it lexicographically
$$C_4 = (\delta^{(11)}, \delta^{(17)}, \delta^{(23)}, \delta^{(37)})$$
where $\delta^{(11)} = (2,5,4,3,1)$, $\delta^{(17)} = (3,4,5,1,2)$, $\delta^{(23)} = (4,1,2,5,3)$ and $\delta^{(37)} = (5,3,1,2,4)$. The corresponding Latin square is $L_5^{(1)} = I_n + 2\phi(\delta^{(2)}) + 3\phi(\delta^{(17)}) + 4\phi(\delta^{(30)}) + 5\phi(\delta^{(36)})$, that is
$$L_5^{(1)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 5 & 3 & 2 \\ 5 & 4 & 1 & 2 & 3 \\ 3 & 5 & 2 & 1 & 4 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix} \qquad (7)$$

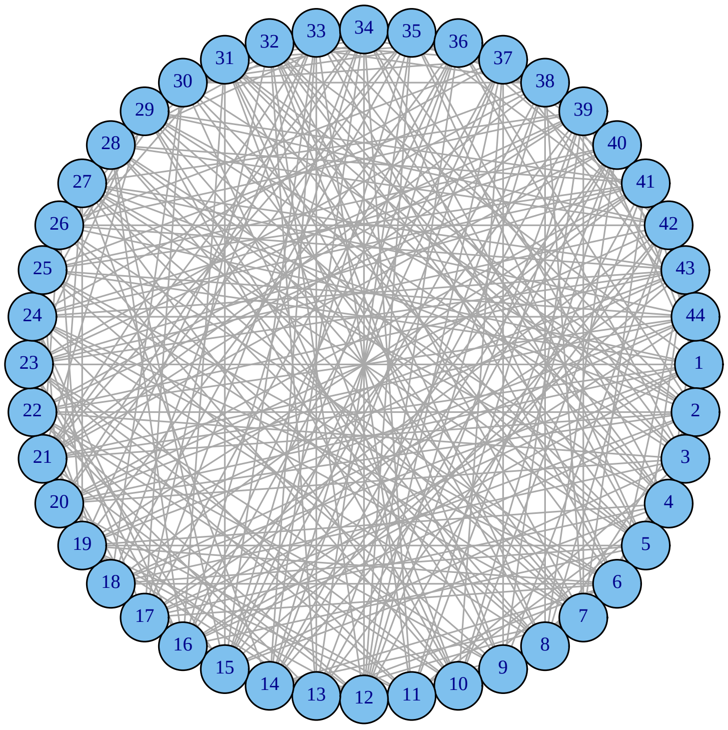

FIG 1. *The graph $\mathcal{G}_5$.*

5. we finally get L_5 by randomly choosing one permutation σ for the symbols $2, \ldots, 5$, $\sigma = (4, 3, 2, 5)$, and one permutation γ for the columns $1, \ldots, 5$, $\gamma = (3, 1, 2, 4, 5)$

$$L_5 = \begin{pmatrix} 3 & 1 & 4 & 2 & 5 \\ 5 & 2 & 1 & 3 & 4 \\ 1 & 5 & 2 & 4 & 3 \\ 4 & 3 & 5 & 1 & 2 \\ 2 & 4 & 3 & 5 & 1 \end{pmatrix}. \tag{8}$$

3.2. Sudoku designs

The algorithm remains the same apart from the substitution of the graph $\mathcal{G}_n$ with $\tilde{\mathcal{G}}_n$ and by limiting the permutations of step (5) to the permutations of the rows within band and of the columns within stacks. For example, for the case $n = 2^2$ we find three maximum cliques of $\tilde{\mathcal{G}}_4$. By randomly choosing one permutation of the symbols $2, 3, 4$ among the six available, and one S-matrix to be used as $\tilde{P}^{(1)}$, among the sixteen available, we can randomly generate one Sudoku design from the total of 288, ($\#\mathcal{S}_4 = 288$).

TABLE 1
Number of derangements of $(1, \dots, n)$, $n \le 9$

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
1	0	1	2	9	44	265	1854	14833	133496

3.3. Computational aspects

We ran the algorithm using a standard laptop (CPU Intel Core i7-2620M CPU 2.70 GHz 2.70 GHz, RAM 8 Gb). We were able to solve the problems corresponding to the orders up to $n = 7$ for which we found $16, 942, 080$ cliques. For $n = 7$ we used Cliquer [8] to find all the cliques. Taking into account symbol and column permutations our algorithm was able to extract uniformly at random a Latin square of order 7 among all the order 7 Latin squares that are $7!6!16, 942, 080 = 61, 479, 419, 904, 000$.

For Latin squares the number of nodes of $\mathcal{G}_n$ coincides with the number of derangements (see Table 1). For Sudoku designs the numbers of Sudoku-derangements are 7 for $n = 4$ and $17, 972$ for $n = 9$.

If n becomes large with respect to the available computational resources it is possible to replace the graph $\mathcal{G}_n$ $(\tilde{\mathcal{G}}_n)$ with a random subgraph $\mathcal{A}_n^k$ $(\tilde{\mathcal{A}}_n^k)$ of it, where k denotes the number of the selected nodes. We point out that, if we take one clique at random from those of the subgraph $\mathcal{A}_n^k$ $(\tilde{\mathcal{A}}_n^k)$, the distribution from which we are sampling is not uniform. Nonetheless, this approach can be useful for selecting the starting point of the algorithm described in [7], which is based on moves between different designs. We experimented with this approach for the 9×9 Sudoku design, which is the most common structure for the popular Sudoku puzzle. We randomly chose 809 Sudoku derangements among the $17, 972$ available. The subgraph has $112, 579$ edges. Its largest cliques have dimensions equal to 8 and are 73. By randomly choosing one clique, one permutation of the symbols $2, \dots, 9$ and one Sudoku matrix we can generate the Sudoku $S_9 \in \mathcal{S}_9$

$$S_9 = \begin{pmatrix} 1 & 3 & 4 & 5 & 7 & 6 & 2 & 9 & 8 \\ 8 & 7 & 2 & 1 & 4 & 9 & 6 & 3 & 5 \\ 6 & 9 & 5 & 3 & 2 & 8 & 1 & 7 & 4 \\ 7 & 1 & 9 & 8 & 5 & 3 & 4 & 2 & 6 \\ 2 & 8 & 6 & 7 & 1 & 4 & 9 & 5 & 3 \\ 4 & 5 & 3 & 6 & 9 & 2 & 8 & 1 & 7 \\ 3 & 4 & 1 & 9 & 6 & 7 & 5 & 8 & 2 \\ 5 & 2 & 8 & 4 & 3 & 1 & 7 & 6 & 9 \\ 9 & 6 & 7 & 2 & 8 & 5 & 3 & 4 & 1 \end{pmatrix}$$

It is worth noting that recent advances in software for huge graph analysis (millions of nodes) make it possible to manage problems that are extremely interesting from a practical point of view.

4. Conclusion

This paper presented a simple algorithm for uniform random sampling from the population of Latin squares and Sudoku designs. The algorithm is based on the largest cliques of proper graphs and has been implemented in SAS. The code exports the graph in a format that can be used by other software, like Cliquer, [8]. The algorithm could be run using the entire graph $\mathcal{G}_n$ up to an order n equal to 7 on a standard desktop machine.

Future research will aim at testing the algorithm for higher orders. Recent advances in graph analytics on huge graphs such as those naturally arising in social sciences (see e.g. [9] for an overview on the subject), make this objective feasible and challenging at the same time. In this case, as a first approach the algorithm could remain the same, i.e. based on sampling the largest cliques of the proper graphs, $\mathcal{G}_n$ for Latin squares or $\tilde{\mathcal{G}}_n$ for Sudoku designs.

Other approaches that could be used for higher orders rely on the replacement of $\mathcal{G}_n$ ($\tilde{\mathcal{G}}_n$) with a random subgraph $\mathcal{A}_n^k$ ($\tilde{\mathcal{A}}_n^k$). As we discussed in the paper, if we take one clique at random from those of the subgraph $\mathcal{A}_n^k$ ($\tilde{\mathcal{A}}_n^k$), the distribution from which we are sampling is not uniform. Nonetheless, this approach can be useful for selecting the starting point of the algorithm described in [7], which is based on moves between different designs. Another possibility that could be investigated is based on building a (non-uniform) distribution of the subgraphs so that the random sampling of the largest cliques becomes uniform. The order of the nodes of $\mathcal{G}_n$ ($\tilde{\mathcal{G}}_n$) could be used to build such a distribution of the subgraphs.

5. Acknowledgements

The author wishes to thank the Editor and Referee for the accurate revision that helped produce a clearer version of the work.

Supplementary Material

Supplement to "Random Latin squares and Sudoku designs generation"
(doi: 10.1214/14-EJS913SUPP; .zip).

References

[1] Bailey, R. A., Cameron, P. J. and Connelly, R. (2008). Sudoku, gerechte designs, resolutions, affine space, spreads, reguli, and Hamming codes. *Amer. Math. Monthly* **115** 383–404. MR2408485

[2] Behrens, W. (1956). Feldversuchsanordnungen mit verbessertem Ausgleich der Bodenunterschiede. *Zeitschrift für Landwirtschaftliches Versuchsund Untersuchungswesen* **2** 176–193.

[3] Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal* **Complex Systems** 1695.

[4] Dahl, G. (2009). Permutation matrices related to Sudoku. *Linear Algebra and Its Applications* **430** 2457–2463. MR2508304

[5] Fontana, R. (2011). Fractions of permutations. An application to Sudoku. *Journal of Statistical Planning and Inference* **141** 3697–3704. MR2823640

[6] Fontana, R. (2014). Supplement to "Random Latin squares and Sudoku designs generation". DOI: 10.1214/14-EJS913SUPP.

[7] Jacobson, M. T. and Matthews, P. (1996). Generating uniformly distributed random Latin squares. *Journal of Combinatorial Designs* **4** 405–437. MR1410617

[8] Niskanen, S. and Östergård, P. R. (2003). *Cliquer User's Guide: Version 1.0*. Helsinki University of Technology.

[9] Shao, B., Wang, H. and Xiao, Y. (2012). Managing and mining large graphs: Systems and implementations. In *Proceedings of the 2012 International Conference on Management of Data* 589–592. ACM.

[10] Yates, F. (1933). The formation of Latin squares for use in field experiments. *Empire Journal of Experimental Agriculture* **1** 235–244.

[11] (2012). SAS/OR(R) 12.1 User's Guide: Network Optimization Algorithms.