UNIFORM GENERATION OF RANDOM LATIN RECTANGLES

Brendan D. McKay

Computer Science Department, Australian National University,

GPO Box 4, ACT 2601, Australia

Nicholas C. Wormald

Department of Mathematics and Statistics, University of Auckland,

Private Bag, Auckland, New Zealand

Abstract.

We show how to generate $k \times n$ Latin rectangles uniformly at random in expected time $O(nk^3)$, provided $k = o(n^{1/3})$. The algorithm uses a switching process similar to that recently used by us to uniformly generate random graphs with given degree sequences.

1. Introduction.

Let $1 \le k \le n$. A $k \times n$ *Latin rectangle* is a $k \times n$ matrix $A = (a_{ij})$, with entries from $N = \{1,2,\ldots,n\}$, such that no two entries in the same row or in the same column are equal. It follows that each row of A forms a permutation of N.

There is a considerable body of literature devoted to Latin rectangles, but little or nothing that specifically deals with random generation. The paper [1] contains a survey of the relevant enumeration work, and also some limited theory on the structure of random Latin rectangles obtained by analysing the effect of interchanging pairs of entries in the same row. In the current paper, we will use a different switching operation in conjunction with an accept-reject strategy to perform uniform random generation. This approach was inspired by the algorithm in [2], which uses a very similar method to generate random graphs with a given degree sequence.

To our knowledge, the only algorithm previously known for this problem is the following, which is implicit in many enumeration papers.

```
repeat
    for $i$ from 1 to $k$ do
        assign a random permutation of $N$ to $a_{i1}, a_{i2}, \ldots, a_{in}$.
    endfor
until no pair of entries in the same column of $A$ are equal
```

Unfortunately, the expected running time of this algorithm is approximately exponential in k^2, making it unusable except for very small k. This follows from well-known enumeration results (see [1]).

2. Switchings and their analysis.

Define $K = \{1, 2, \ldots, k\}$. Let $M(k,n)$ be the set of all $k \times n$ matrices over N such that no two entries in the same row are the same. For $i \in K$ and $j \in N$, define $C_j = \{(t,j) : t \in K\}$. Further, for $A \in M(k,n)$ and $E \subseteq K \times N$, define $A[E] = \{a_{ij} : (i,j) \in E\}$. Thus, for example, $A[C_2]$ is the set of entries in the second column of A.

If $A = (a_{ij}) \in M(k,n)$, a *conflict* in A is a pair $(\{i_1, i_2\}, j)$ such that $i_1 \neq i_2$ and $a_{i_1 j} = a_{i_2 j}$. Two distinct conflicts $(\{i_1, i_2\}, j)$ and $(\{i'_1, i'_2\}, j')$ are said to *overlap* if $j = j'$ and $\{i_1, i_2\} \cap \{i'_1, i'_2\} \neq \emptyset$. A *directed conflict* is a 3-tuple (i_1, i_2, j) such that $(\{i_1, i_2\}, j)$ is a conflict.

Lemma 2.1. *Choose a random member A of $M(k,n)$ for $k = o(n^{1/3})$. Then, with probability at least $\frac{1}{2} + o(1)$, A has at most k^2 conflicts and no overlapping conflicts.*

Proof. The expected number of conflicts is easily seen to be exactly $\binom{k}{2}$, while the expected number of pairs $(\{i_1, i_2, i_3\}, j)$ where i_1, i_2, i_3 are distinct and $a_{i_1 j} = a_{i_2 j} = a_{i_3 j}$ is $\binom{k}{3}/n$. The claim follows immediately. ∎

For $A \in M(k,n)$, define $d(A)$ to be the maximum number of conflicts which involve any one row of A.

Lemma 2.2. *Choose a random member A of $M(k,n)$ for $k = o(n^{1/3})$. Then, with probability at least $1 - O(n^{-1/6})$, $d(A) < 5k + \ln n$.*

Proof. Let $D = \lceil 5k + \ln n \rceil$. Consider the number T of sets of D conflicts involving the first row. There are at most $(kD)^D/D!$ available positions for conflicts, and each one occurs (subject to any subset of the others occurring) with probability at most $(n-D)^{-D}$. Therefore, the expectation of T is bounded above by $(kD)^D/(D!(n-D)^D) = O(1)(ek/D)^D = O(1)(e/5)^{\ln n} = O(n^{-1/2})$. It follows that the probability of D or more conflicts in any one row is $O(kn^{-1/2}) = O(n^{-1/6})$. ∎

For $c \geq 0$, define $M_c = M_c(k,n)$ to be the set of all members of $M(k,n)$ which have at exactly c conflicts and no overlapping conflicts. For $A \in M(k,n)$ define $\mathrm{Sw}(A)$ to be the set of all quintuples $(i_1, i_2; j_1, j_2, j_3)$ such that $i_1, i_2 \in K$ are distinct, $j_1, j_2, j_3 \in N$ are distinct, and conditions X_1–X_6 are satisfied, where $y = a_{i_1 j_1}$, $u = a_{i_1 j_2}$ and $v = a_{i_1 j_3}$:

X_1: $a_{i_2 j_1} = y$;

X_2: $u \notin A[C_{j_2} - \{(i_1, j_2)\}]$;

X_3: $v \notin A[C_{j_3} - \{(i_1, j_3)\}]$;

X_4: $y \notin A[C_{j_2}]$;

X_5: $u \notin A[C_{j_3}]$;

X_6: $v \notin A[C_{j_1}]$.

If $\sigma = (i_1, i_2; j_1, j_2, j_3) \in \mathrm{Sw}(A)$, then the *switching* $\mathrm{sw}(\sigma)$ is the operation mapping A onto the matrix $A' = (a'_{ij})$ with the same entries as A except that $a'_{i_1 j_1} = v$, $a'_{i_1 j_2} = y$ and $a'_{i_1 j_3} = u$. Note that $A' \in M_{c-1}$ if and only if $A \in M_c$. A pictorial representation of $\mathrm{sw}(\sigma)$ appears in Figure 1.

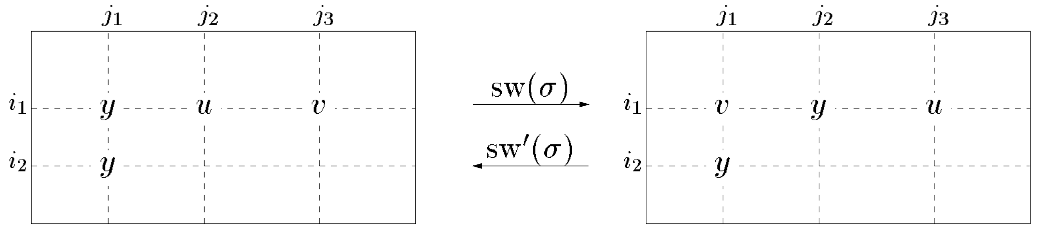

Figure 1. The switching $\mathrm{sw}(\sigma)$ and its reverse $\mathrm{sw}'(\sigma)$.

For $A' \in M(k, n)$, define $\mathrm{Sw}(A')$ to be the set of all quintuples $(i_1, i_2; j_1, j_2, j_3)$ such that $i_1, i_2 \in K$ are distinct, $j_1, j_2, j_3 \in N$ are distinct, and conditions Y_1–Y_7 are satisfied, where $y = a'_{i_1 j_2}$, $u = a'_{i_1 j_3}$ and $v = a'_{i_1 j_1}$:
Y_1: $a'_{i_2 j_1} = y$;
Y_2: $v \notin A'[C_{j_1} - \{(i_1, j_1)\}]$;
Y_3: $y \notin A'[C_{j_1} - \{(i_2, j_1)\}]$;
Y_4: $y \notin A'[C_{j_2} - \{(i_1, j_2)\}]$;
Y_5: $u \notin A'[C_{j_3} - \{(i_1, j_3)\}]$;
Y_6: $u \notin A'[C_{j_2}]$;
Y_7: $v \notin A'[C_{j_3}]$.
If $\sigma = (i_1, i_2; j_1, j_2, j_3) \in \mathrm{Sw}(A')$, then the *reverse switching* $\mathrm{sw}'(\sigma)$ is the operation mapping A' onto the matrix $A = (a_{ij})$ with the same entries as A' except that $a_{i_1 j_1} = y$, $a_{i_1 j_2} = u$ and $a_{i_1 j_3} = v$.

It is easy to see that $\mathrm{sw}(\sigma)$ takes A onto A' if and only if $\mathrm{sw}'(\sigma)$ takes A' onto A. Furthermore, the conditions X_1–X_6 are consistent with Y_1–Y_7 in the sense that $\sigma \in \mathrm{Sw}(A)$ if and only if $\sigma \in \mathrm{Sw}'(A')$.

Lemma 2.3. *Let $A \in M_c$, where $c \geq 1$, and let $d = d(M)$. Then*

$$2cn^2 - 6ckn - 4cdn \leq |\mathrm{Sw}(A)| \leq 2cn^2. \tag{1}$$

Proof. Consider finding all the quintuples $(i_1, i_2; j_1, j_2, j_3)$. We can choose (i_1, i_2, j_1) in c ways, then j_2 in at most $n-1$ ways and then j_3 in at most $n-2$ ways. This gives us an upper bound of $2c(n-1)(n-2)$, which gives us the right inequality of (1).

To obtain a lower bound, we subtract upper bounds on the numbers of quintuples we have just included, but which fail one of X_2–X_6. (X_1 is already satisfied.) In each case the argument is elementary; we merely list the bounds here.

X_2: $2c(d-1)(n-2)$;
X_3: $2c(d-1)(n-2)$;
X_4: $2c(k-2)(n-2)$;
X_5: $2c(k-1)(n-1)$;
X_6: $2c(k-2)(n-2)$;
In total, we find a lower bound of

$$2c(n-1)(n-2) - 4c(k-2)(n-2) - 2c(k-1)(n-1) - 4c(d-1)(n-2),$$

which implies the left inequality of (1). ∎

Lemma 2.4. *Let $A' \in M_{c-1}$, where $c \geq 1$. Then*

$$k(k-1)n^2 - 2k^3 n - 8ckn \leq |\mathrm{Sw}'(A')| \leq k(k-1)n^2. \tag{2}$$

Proof. This follows along the lines of the proof of Lemma 2.3. To get an upper bound on the number of $(i_1, i_2; j_1, j_2, j_3) \in \mathrm{Sw}'(A')$, note that we can choose (i_1, j_1) in kn ways, then i_2 in at most $k-1$ ways (which then determines j_2 by Y_1), then j_3 in at most $n-2$ ways. This gives a total of at most $k(k-1)n(n-2)$, implying the right inequality of (2).

Conditions Y_2–Y_7 each eliminate some of these choices from consideration; we list upper bounds for each.
Y_2: $2(c-1)(k-2)(n-2)$;
Y_3: $2(c-1)(k-2)(n-2)$;
Y_4: $2(c-1)(k-2)(n-2)$;
Y_5: $2(c-1)(k-1)(n-1)$;
Y_6: $k(k-1)^2 n$;
Y_7: $k(k-1)^2 n$;
Overall, we have a lower bound of

$$k(k-1)n(n-2) - 6(c-1)(k-2)(n-2) - 2(c-1)(k-1)(n-1) - 2k(k-1)^2 n,$$

which implies the left inequality of (1). ∎

We mention in passing that Lemmas 2.3 and 2.4 provide a very simple derivation of the asymptotic number of $k \times n$ Latin rectangles for $k = o(n^{1/3})$, a result due first to Yamamoto [3]. However, the estimate obtained is far weaker than that attainable by other methods such as those used in [1].

3. The algorithm.

The approach we will take towards generating Latin rectangles will be to generate members of $M(k,n)$ and then apply switching operations until all conflicts have been removed. In order to produce a uniform distribution, we will use an accept-reject strategy; the closeness of the upper and lower bounds in Lemmas 2.3 and 2.4 will ensure a respectable average running time.

Define $S(c) = k(k-1)n^2 - 2k^3n - 8ckn$.

```
function LATIN($k,n$)  — generate a random $k \times n$ Latin rectangle
    repeat
        repeat
            $A :=$ random member of $M(k,n)$
        until $A$ has at most $k^2$ conflicts and no overlapping conflicts
        $c :=$ the number of conflicts in $A$
        $rejected := false$
        while $c > 0$ and not $rejected$ do
            $(i_1, i_2, j_1) :=$ a random directed conflict of $A$
            $j_2 :=$ a random member of $N - \{j_1\}$
            $j_3 :=$ a random member of $N - \{j_1, j_2\}$
            if $(i_1, i_2; j_1, j_2, j_3) \in \mathrm{Sw}(A)$ then
                $A :=$ the result of applying $\mathrm{sw}(\sigma)$ to $A$
                with probability $1 - S(c)/|\mathrm{Sw}'(A)|$ do $rejected := true$ endwith
            else
                $rejected := true$
            endif
            $c := c - 1$
        endwhile
    until not $rejected$
    return $A$
end
```

Theorem 3.1. *If $1 \le k \le n$ and $S(k^2) > 0$, function* LATIN *generates $k \times n$ Latin rectangles uniformly at random.*

Proof. The condition $S(k^2) > 0$ ensures that LATIN will eventually return, with probability one. This follows from Lemma 2.3.

Let us say that a rectangle A is *accepted* by LATIN if A is the value of the variable of that name when the inner **repeat** loop completes or when an iteration of the **while** loop ends with $rejected = false$.

Suppose that $A_c, A_{c-1}, \ldots$ is the sequence of rectangles accepted during a single iteration of the outer **repeat** loop, where $A_c \in M_c$. It is immediately clear that (given c), A_c is chosen uniformly at random from M_c.

As an induction hypothesis, suppose that, for some i ($c \geq i \geq 1$), the probability that $A_i \in M_i$ is accepted is independent of A_i. We will show that the same is true of the probability that $A_{i-1} \in M_i$ is accepted. Take an arbitrary $B' \in M_{i-1}$ and let $\mathcal{B}(B')$ be the set of all B such that B' can be obtained from B by applying $\mathrm{sw}(\sigma)$ for some $\sigma \in \mathrm{Sw}(B)$. Clearly, $\mathcal{B}(B') \subseteq M_i$ and $|\mathcal{B}(B')| = |\mathrm{Sw}'(B')|$. If p is the (uniform) probability of accepting each $B \in M_i$, then the probability of accepting B' is

$$\sum_{B \in \mathcal{B}(B')} \frac{p}{2i(n-1)(n-2)} \frac{S(i)}{|\mathrm{Sw}'(B')|} = \frac{pS(i)}{2i(n-1)(n-2)},$$

which is independent of B'. The theorem is now immediate. ∎

4. The average complexity of the algorithm.

We will now show that function LATIN can be implemented so that the average running time is $O(nk^3)$ if $k = o(n^{1/3})$. We will assume a machine model in which elementary operations on integers of $O(\log n)$ digits require $O(1)$ time, and in which all variables have initial (arbitrary) values.

An important parameter is the average number of executions of the outer **repeat** loop.

Lemma 4.1. *If $k = o(n^{1/3})$, the expected number of iterations of the outer* **repeat** *loop of function* LATIN *during one invocation is* $1 + o(1)$.

Proof. Let A be the matrix produced by the inner **repeat** loop. Suppose A has c conflicts; by design $c \leq k^2$.

By Lemma 2.2, there is a probability of $1 - o(1)$ that $d < 5k + \ln n$ where $d = d(A)$; suppose that is the case. Then the probability of flag *rejected* being set during the exection of the **while** loop is at most

$$\sum_{i=1}^{c} \Bigl(1 - \frac{|\mathrm{Sw}(A_i)|}{2i(n-1)(n-2)} \frac{S(i)}{|\mathrm{Sw}'(A_{i-1})|}\Bigr),$$

where the sequence $A_c, A_{c-1}, \ldots$ is defined as in the proof of Theorem 3.1. However, Lemmas 2.3 and 2.4 imply that

$$\begin{aligned}1 - \frac{|\mathrm{Sw}(A_i)|}{2i(n-1)(n-2)} \frac{S(i)}{|\mathrm{Sw}'(A_{i-1})|}&\\ &\leq 1 - \frac{(k(k-1)n^2 - 2k^3 n - 8ikn)(2in^2 - 6ikn - 4idn)}{2ik(k-1)n^4}\\ &= O\Bigl(\frac{k+d}{n} + \frac{i}{kn}\Bigr),\end{aligned}$$

from which the lemma follows. ∎

The number of conflicts, and the presence of overlapping conflicts, in a matrix $A \in M(k,n)$ can be determined easily in $O(nk^2)$ time. Thus, by Lemma 2.1, at most $O(nk^2)$ time is required for the completion of the inner **repeat** loop for each execution of the outer **repeat** loop.

By construction, the **while** loop is executed at most k^2 times. To make the execution of this loop efficient, we introduce a number of auxiliary arrays. Recall that $A = (a_{ij})$.

$C[i]$ is the i-th conflict in A $(1 \le i \le c)$.

$I[i,j]$ is the value t such that $a_{it} = j$ $(i \in K, j \in N)$.

$V[j,t]$ is the number of times t occurs in column j $(j,t \in N)$.

Array C can be constructed for the initial A in time $O(nk^2)$ and updated after each switching in time $O(1)$. The same holds true for the array I.

For the array V, we must take care to avoid an $O(n^2)$ initialization phase. This can be achieved by implementing V as a pair of arrays $V_1[j,t]$ $(j,t \in N)$ and $V_2[s,u]$ $(1 \le s \le m,\ 1 \le u \le 3)$, where m is the sum of the numbers of different entries in each column. The values $(V_2[s,1], V_2[s,2], V_2[s,3])$ are all the triples $(j,t,V[j,t])$ for which $V[j,t] > 0$, in no particular order. For each such s, $V_1[j,t] = s$; other entries of V_1 are arbitrary. Since V_1 doesn't need initialization, the arrays V_1 and V_2 can be constructed from A in $O(nk)$ time. As each switching is performed, they can be updated in $O(1)$ time. Further, any entry of V can be determined or modified in $O(1)$ time.

Given the array C, we can choose a random directed conflict in $O(1)$ time. The test "$(i_1,i_2;j_1,j_2,j_3) \in \mathrm{Sw}(A)$" can be performed in $O(1)$ time also, using the array V. The only significant contribution to the running time that we haven't accounted for is the computation of $|\mathrm{Sw}'(A)|$.

For given A, define $Q = Q(A)$ to be the set of all quintuples $(i_1,i_2;j_1,j_2,j_3)$, with distinct $i_1,i_2 \in K$ and distinct $j_1,j_2,j_3 \in N$. The events $Y_1,\ldots,Y_7$ defined in Section 2 can be identified as subsets of Q. For any $Y \subseteq Q$, define $\bar{Y} = Q - Y$. Then we have

$$\begin{aligned}
|\mathrm{Sw}'(A)| &= |Y_1 \cap Y_2 \cap Y_3 \cap Y_4 \cap Y_5 \cap Y_6 \cap Y_7| \\
&= |Y_1| - |Y_1 \cap \bar{Y}_2| - |Y_1 \cap Y_2 \cap \bar{Y}_3| \\
&\quad - |Y_1 \cap Y_2 \cap Y_3 \cap \bar{Y}_4| - |Y_1 \cap Y_2 \cap Y_3 \cap Y_4 \cap \bar{Y}_5| \\
&\quad - |Y_1 \cap Y_2 \cap Y_3 \cap Y_4 \cap Y_5 \cap \bar{Y}_6| - |Y_1 \cap Y_2 \cap Y_3 \cap Y_4 \cap Y_5 \cap Y_6 \cap \bar{Y}_7|. \quad (3)
\end{aligned}$$

Let $n_1,\ldots,n_7$ denote the seven cardinalities on the right side of (3), and suppose that A has c conflicts. Then, recalling that A has no overlapping conflicts, we have

$$n_1 = n(n-2)k(k-1) - 2c(n-2)$$

and

$$n_2 = 2c(k-2)(n-2).$$

To compute n_3, take each directed conflict (i_2, i_3, j_1), and each $i_1 \in K - \{i_2, i_3\}$, and test if Y_2 is satisfied using the array V. The value of n_3 is then $n-2$ times the number of successful tests. This takes $O(k^3)$ time.

To compute n_4, take each directed conflict (i_1, i_3, j_2), and each $i_2 \in K - \{i_1, i_3\}$, and test if $Y_2 \cap Y_3$ is satisfied for $j_1 = I[i_2, a_{i_1 j_2}]$. The value of n_4 is then $n-2$ times the number of successful tests. This takes $O(k^3)$ time also.

To compute the values of n_5, n_6 and n_7, we need a more complex approach. For the initial A (i.e., the one which passes the inner **repeat** loop) these values can be computed in time $O(nk^3)$ time by systematically testing each of the possibilities. Then, as each switching is done, the change to these values can be computed in $O(nk)$ time. To see this, notice that whether or not a particular quintuple $(i_1, i_2; j_1, j_2, j_3)$ counts towards these values is unchanged by a switching unless at least one of the columns j_1, j_2, j_3 is changed. Only three columns are changed, so there are only $O(nk)$ quintuples whose membership of $Y_1 \cap \bar{Y}_5$ might be affected. The corresponding bounds for $Y_1 \cap \bar{Y}_6$ and $Y_1 \cap \bar{Y}_7$ are both $O(k^3)$. In each case, we can look at each possibility and test membership of $Y_2 \cap Y_3 \cap \cdots$ in $O(1)$ time using array V.

We conclude that all the computations of $|\mathrm{Sw}'(A)|$ required for one execution of the outer **repeat** loop can be performed in $O(nk^3)$ time. Putting this together with the other bounds we have found, we conclude that the following theorem is true.

Theorem 4.2. *Let $k = o(n^{1/3})$. Then function LATIN can be implemented so that the expected running time is $O(nk^3)$.* ∎

References.

[1] C. D. Godsil and B. D. McKay, Asymptotic enumeration of Latin rectangles, *J. Combinatorial Theory, Ser. B*, **48** (1990) 19–44.

[2] B. D. McKay and N. C. Wormald, Uniform generation of random regular graphs of moderate degree, *J. Algorithms*, to appear.

[3] K. Yamamoto, On the asymptotic number of Latin rectangles, *Japan J. Math.* **21** (1951) 113–119.