JOURNAL OF ALGORITHMS 11, 52–67 (1990)

Uniform Generation of Random Regular Graphs of Moderate Degree

BRENDAN D. MCKAY

Computer Science Department, Australian National University, G.P.O. Box 4, ACT 2601, Australia

AND

NICHOLAS C. WORMALD

Department of Mathematics and Statistics, University of Auckland, Private Bag, Auckland, New Zealand

Received August 17, 1988; revised March 7, 1989

We show how to generate k-regular graphs on n vertices uniformly at random in expected time $O(nk^3)$, provided $k = O(n^{1/3})$. The algorithm employs a modification of a switching argument previously used to count such graphs asymptotically for $k = o(n^{1/3})$. The asymptotic formula is re-derived, using the new switching argument. The method is applied also to graphs with given degree sequences, provided certain conditions are met. In particular, it applies if the maximum degree is $O(|E(G)|^{1/4})$. The method is also applied to bipartite graphs. © 1990 Academic Press, Inc.

1. INTRODUCTION

Random regular graphs have come under ever increasing scrutiny in recent years. However, it is not easy to generate k-regular graphs on n vertices uniformly at random. It is known how to do this for small k in expected time $O(e^{k^2/2}nk)$ per graph, using a procedure which does not necessarily terminate (see Wormald [5] or Bollobás [1]); but even for $k \approx \log n$ this is not polynomial expected time. If one insists on an algorithm which always terminates, the picture is even worse; it can be done [5] for $k = 3$ and 4 but already the algorithm is very complicated. On the other hand, one can slacken the uniformity constraint slightly and ask for an almost uniform probability distribution. Sinclair and Jerrum [4] were

0196-6774/90 $3.00

successful at generating random graphs of this type with given degrees in polynomial time, as long as the degrees are bounded above by $O(m^{1/4})$, where m is the number of edges. For this, they employed Markov processes and asymptotic enumeration results obtained by McKay [2] using switchings.

Our aim here is to show how to generate graphs with given degrees uniformly at random in polynomial expected time. Our result applies to a slightly wider range of degree sequences than Sinclair and Jerrum's. To do this we combine features of the basic method of the algorithm for generating k-regular graphs in [5] with a type of switching related to that in [2]. This new type of switching also enables extension of the asymptotic enumeration results (see McKay and Wormald [3]).

Our model of a graph G with vertex degrees $k_1, \ldots, k_n$ is a set of $M = \sum k_i$ *points* arranged in *cells* of size $k_1, k_2, \ldots, k_n$. We take a partition (called a *pairing*) P of the M points into $\frac{1}{2}M$ parts (called *pairs*) of size 2 each. The *degrees* of P are $k_1, \ldots, k_n$. The vertices of G are identified with the cells and the edges with the pairs; each edge of G joins the vertices in which the points of the corresponding pair lie. A *loop* of P is a pair whose two points lie in the same cell. A *multiple pair* is a maximal set of $j \geq 2$ pairs each involving the same two cells; this is a *double pair* if $j = 2$, a *triple pair* if $j = 3$, and a *double loop* if the two cells are the same. The *mate* of a point is the other point in its pair.

If the pairing has multiple pairs then G is strictly a *multigraph* rather than a graph; we also forbid loops in a graph. For $j \geq 2$, a *j-path* is a sequence $p_1, \ldots, p_{2j}$ of points such that p_{2i} and p_{2i+1} are distinct but in the same cell, for $i = 1, \ldots, j-1$. Note that each non-loop double pair contains four distinct 2-paths, two beginning at each cell involved.

We make use of the following two operations on a pairing:

I. *l-Switching*

Take pairs $\{p_1, p_6\}, \{p_2, p_3\}, \{p_4, p_5\}$, where $\{p_2, p_3\}$ is a loop, and p_1, p_2, p_3, p_4, p_5, and p_6 are in five different cells. Replace these pairs by $\{p_1, p_2\}, \{p_3, p_4\}, \{p_5, p_6\}$. In this operation, none of the pairs created or destroyed is permitted to be part of a multiple pair (see Fig. 1).

II. *d-Switching*

Take pairs $\{p_1, p_5\}, \{p_2, p_6\}, \{p_3, p_7\}, \{p_4, p_8\}$, where p_2 and p_3 are in the same cell, as are p_6 and p_7, but the cells containing $p_1, p_2, p_4, p_5, p_6, p_8$ are all distinct. Replace these pairs by $\{p_1, p_2\}, \{p_3, p_4\}, \{p_5, p_6\}, \{p_7, p_8\}$. Note that these form two 2-paths. In this operation, none of the pairs created or destroyed is permitted to be part of a multiple pair, except that $\{p_2, p_6\}, \{p_3, p_7\}$ form a double pair (see Fig. 2).

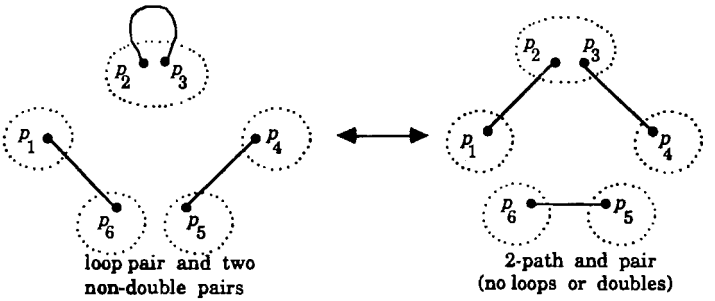

FIG. 1. *l*-switching.

A *forward* *l*-switching is an *l*-switching as described, and a *backward* *l*-switching is the reverse operation. We use the same convention for *d*-switchings. Note that a forward *l*-switching always reduces the number of loops by 1 and does not create or destroy double pairs. Similarly, a forward *d*-switching reduces the number of double pairs by 1 and neither creates nor destroys loops.

In the next section, we analyse random pairings and the number of ways that the switching operations can be carried out in pairings with given numbers of loops and double pairs. From this, McKay's formula for the asymptotic number of *k*-regular graphs is re-derived in Section 3, and in Section 4 we give a procedure DEG for generating degree-constrained graphs uniformly at random. In Section 5, we show how to reduce the asymptotic average-case time complexity of DEG in the case of regular graphs. Finally, in Section 6 we discuss the modifications required to apply the same method to bipartite graphs with given degrees. Note that these are

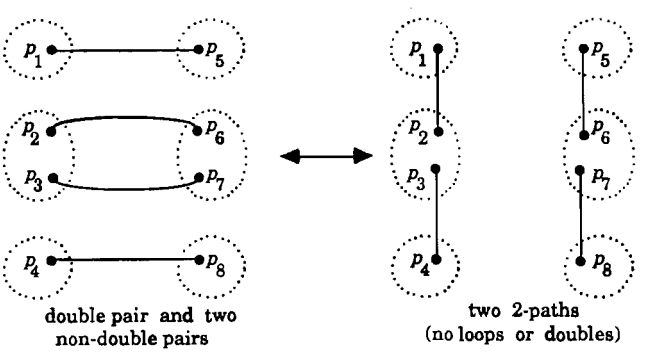

FIG. 2. *d*-switching.

equivalent to $(0,1)$-matrices with given row and column sums, whereas graphs with given degrees correspond to symmetric $(0,1)$-matrices with zero diagonal and given row sums.

2. Preliminary Results

In this section, we consider a pairing P with M points and degrees $k_1, \ldots, k_n$, with $k_i \le k = k(n)$ for $i = 1, \ldots, n$. The first four lemmas, 1 to 3′, refer to such a pairing P chosen uniformly at random. The notation o, O, and $\sim$ refers to n tending to ∞, as does $\to$ when used in connexion with functions, and our results are uniform over all sequences $k_1, \ldots, k_n$ as above, provided $M \to \infty$. We use E to denote expectation, and put

$$M_2 = M_2(k_1, \ldots, k_n) = \sum_{i=1}^{n} k_i(k_i - 1).$$

Lemma 1. *The probability of t given pairs occurring in P is at most $(M - 2t)^{-t}$, which is asymptotic to M^{-t} for t fixed.*

Proof. To be precise, the probability is

$$\frac{[M/2]_t 2^t}{[M]_{2t}},$$

where $[x]_t$ denotes $x(x-1)\ldots(x-t+1)$. □

Lemma 2. *The probability that P contains at least one triple pair is $O(k^2 M_2^2/M^3)$ and the probability of at least one double loop is $O(k^2 M_2/M^2)$.*

Proof. By Lemma 1, the expected number of triple pairs (other than triple loops) is

$$M^{-3} \sum_{1 \le i < j \le n} 6\binom{k_i}{3}\binom{k_j}{3} \le 6M^{-3}\left(\sum_{1 \le i \le n}\binom{k_i}{3}\right)^2 < \frac{1}{6}M^{-3}(kM_2)^2.$$

Similarly, the expected number of double loops is

$$M^{-2} \sum_{1 \le i \le n} 3\binom{k_i}{4} < k^2 M_2/(8M^2). \qquad \square$$

Let l denote the number of loops, and let d denote the number of double pairs (not in triple pairs) in P. For counting regular graphs, we use the

following:

LEMMA 3. *Let $\omega(n) \to \infty$ with $k^2 + \omega(n) < M/25$. Then*

$$\Pr\{d > k^2 + \omega(n) \quad \text{or} \quad l > 2k + \omega(n)\} = o(1).$$

Proof. By Lemma 1,

$$E\left(\binom{d}{j}\right) \le \frac{1}{j!}\left(\frac{\binom{M}{2}k^2/2}{(M-4j)^2}\right)^j.$$

Setting $j = k^2 + \omega(n)$ and taking it as an integer, we obtain

$$\Pr\{d \ge j\} \le E\left(\binom{d}{j}\right) = O\left(\left(\frac{625j}{1764}\right)^j \Big/ \left(\frac{j}{e}\right)^j\right) = o(1).$$

This, together with a similar computation for l, gives the lemma. □

For the generation of graphs, we will use the following similar result.

LEMMA 3′. *We have*

$$\Pr\left\{d > \left(\frac{M_2}{M}\right)^2 \quad \text{or} \quad l > \frac{M_2}{M}\right\} < \frac{3}{4} + o(1).$$

Proof. By Lemma 1, $E(l) < (1 + o(1))(1/M)\Sigma\binom{k_i}{2} = (1 + o(1))M_2/2M$. Thus, $\Pr\{l > M_2/M\} < \frac{1}{2} + o(1)$. Similarly,

$$\begin{aligned} E(d) &< (1 + o(1))(2/M^2)\Sigma_{1 \le i < j \le n}\binom{k_i}{2}\binom{k_j}{2} \\ &< (1 + o(1))(1/M^2)\left(\Sigma_{1 \le i \le n}\binom{k_i}{2}\right)^2 \\ &= (1 + o(1))(M_2/2M)^2. \end{aligned}$$

So $\Pr\{d > (M_2/M)^2\} < \frac{1}{4} + o(1)$. □

Let $\mathscr{C}_{l,d}$ be the set of pairings with l loops, d double pairs, and no triple pairs or double loops.

LEMMA 4. *Denote an operation taking an element of $\mathscr{C}_{i,j}$ to an element of $\mathscr{C}_{k,l}$ by $\mathscr{C}_{i,j} \to \mathscr{C}_{k,l}$. For each of the following operations, we bound the*

number, m, of ways of applying the operation.

(a) *forward l-switching* $\mathscr{C}_{l,d} \to \mathscr{C}_{l-1,d}$:

$$2lM^2 \geq m \geq 2lM^2\left(1 - O\left(\frac{k^2 + l + d}{M}\right)\right);$$

(b) *backward l-switching* $\mathscr{C}_{l-1,d} \to \mathscr{C}_{l,d}$:

$$MM_2 \geq m \geq MM_2\left(1 - \frac{(k-1)(6(l+2d) + (k-1)l)}{M_2} - \frac{2(k-1)(k+2)}{M}\right);$$

(c) *forward d-switching* $\mathscr{C}_{0,d} \to \mathscr{C}_{0,d-1}$:

$$4dM^2 \geq m \geq 4dM^2\left(1 - O\left(\frac{k^2 + d}{M}\right)\right);$$

(d) *backward d-switching* $\mathscr{C}_{0,d-1} \to \mathscr{C}_{0,d}$:

$$M_2^2 \geq m \geq M_2^2\left(1 - \frac{(k-1)\left(16d + 9(k-1) + 3(k-1)^2\right)}{M_2}\right).$$

Proof. Given a pairing in $\mathscr{C}_{l,d}$ to which a forward l-switching is to be applied, we can choose the points p_1 and p_4 in M ways each, and the point p_2 in $2l$ ways. This determines precisely how the switching is to be applied; for example, the point p_3 is the mate of p_2. Hence the upper bound on m in (a). For some choices of p_1, p_4, and p_2 the switching cannot be performed (for example, if $p_1 = p_4$) or does not yield an element of $\mathscr{C}_{l-1,d}$ due to the creation or destruction of other loops or multiple pairs. These "bad" choices are (over)estimated and subtracted to give the lower bound on m in (a). We will not need a very accurate estimate of this. Similarly, in (b) we can choose the points p_2 and p_3 in M_2 ways, and then p_6 in M ways. Hence the upper bound. For the lower bound, there are three types of things that can go wrong:

(i) a pair chosen might be in a loop or double pair,

(ii) a cell containing p_i for $i \leq 4$ might contain p_5 and p_6,

(iii) the selection might be such that a double pair would be created in the switching. That is, one of three forbidden edges is already present in the graph (one of these is a loop).

We bound the number of possibilities in (i) by $3(2l + 4d)(k - 1)M$, in (ii) by $6M_2(k - 1)$, and in (iii) by $lM(k - 1)^2 + 2M_2(k - 1)^2$. The lower bound follows.

In (c), we choose the points p_2 and p_3 at the same end of a double pair in $4d$ ways, and then points p_1 and p_4 in M ways each, and for the lower bound subtract the number of bad choices as in (a). In (d), we choose p_2 and p_3 in M_2 ways, and p_6 and p_7 similarly. A chosen pair can be a double pair in at most $16d(k - 1)M_2$ ways, a cell can simultaneously contain p_1, p_2, p_3, or p_4 and p_5, p_6, p_7, or p_8 in at most $9M_2k(k - 1)$ ways, and forbidden pairs can be present in at most $3M_2(k - 1)^3$ ways. □

3. Enumeration

We show here that the bounds in Lemma 4 are sufficiently accurate to give a simpler proof of the formula given by McKay [2] for the asymptotic number of k-regular graphs for $k = o(n^{1/3})$. The precise forms of the lower bounds are not even required. We will use this idea elsewhere [3] to extend the asymptotic formula for counting graphs by degree sequence, but here we confine the enumerative discussion to the simpler regular case. We include the proof of the following result because it bears more than passing resemblance to the algorithm given in Section 4 for generating these graphs.

Theorem 1. (McKay [2]). *If $k = o(n^{1/3})$, the number of labelled k-regular graphs on n vertices is*

$$\frac{(nk)!\exp\bigl((1 - k^2)/4\bigr)}{(nk/2)!2^{nk/2}}(1 + o(1))$$

uniformly as $n \to \infty$ with kn even.

Proof. Note that for k-regular graphs, $M = kn$ and $M_2 = k(k - 1)n$. Set $c_{l,d} = |\mathscr{C}_{l,d}|$. Suppose $d < \omega(n) + k^2$ and $l < \omega(n) + k$, where $\omega(n) \to \infty$ arbitrarily slowly. By Lemma 4(a) and (b),

$$\frac{c_{l,d}}{c_{l-1,d}} = \frac{(k - 1)}{2l}\left(1 + O\left(\frac{k + \omega(n)}{n}\right)\right)$$

for $l \geq 1$, and from (c) and (d),

$$\frac{c_{0,d}}{c_{0,d-1}} = \frac{(k - 1)^2}{4d}\left(1 + O\left(\frac{k + \omega(n)}{n}\right)\right).$$

Hence

$$\frac{c_{l,d}}{c_{0,0}} = \frac{1}{d!l!}\left(\frac{k-1}{2}\right)^{2d+l}\left(1+O\left(\frac{k+\omega(n)}{n}\right)\right)^{d+l}$$

$$= \frac{1}{d!l!}\left(\frac{k-1}{2}\right)^{2d+l}(1+o(1)).$$

Thus the sum of $c_{l,d}$ over $0 \le l \le 2k + \omega(n)$ and $0 \le d \le k^2 + \omega(n)$ is $c_{0,0}\exp((k-1)^2/4 + (k-1)/2)$. So by Lemma 3, $\Pr\{\mathscr{G}_{0,0}\} \sim \exp((1 - k^2)/4)$, and the theorem follows. □

4. Generation of Random Graphs with Specified Vertex Degrees

We describe in this section a procedure DEG whose input is n and $k_1, \ldots, k_n$ and output is a random graph on n vertices of degrees $k_1, \ldots, k_n$. It uses two procedures, which eliminate loops and multiple pairs from a random pairing, but which are aborted with a certain probability. Such an abort we denote by **restart**; in this case DEG should be repeated. We assume all the k_i are non-zero to make it easier to state the complexity results. In particular, we have $M \ge n$.

Let B_1 denote the upper bound in Lemma 4(a), B_2 the lower bound in Lemma 4(b), B_3 the upper bound in Lemma 4(c), and B_4 the lower bound in Lemma 4(d). We first have procedures for eliminating loops and double edges.

procedure NOLOOPS (P); [P is a pairing]
  while P has at least one loop **do**
  begin
    obtain a pairing P' by applying a random forward l-switching to P;
    m_1 := the number of ways to apply a forward l-switching to P;
    m_2 := the number of ways to apply a backward l-switching to P';
    restart with probability $1 - (m_1 B_2/m_2 B_1)$; otherwise $P := P'$;
  end;

procedure NODOUBLES (P); [P is a pairing]
  while P has at least one double pair **do**
  begin
    obtain a pairing P' by applying a random forward d-switching to P;
    m_3 := the number of ways to apply a forward d-switching to P;
    m_4 := the number of ways to apply a backward d-switching to P';
    restart with probability $1 - (m_3 B_4/m_4 B_3)$; otherwise $P := P'$;
  end;

Finally, we have the following procedure for generating a random graph G on n vertices of degrees $k_1, \ldots, k_n$:

```
procedure DEG (n, k_1, ..., k_n);
begin
  select a pairing P uniformly at random from the pairings of degrees k_1, ..., k_n;
  if P has any multiple pairs of cardinality greater than 2,
    or a double loop, or more than (M_2/M)^2 double pairs,
    or more than M_2/M loops then restart;
  NOLOOPS (P);
  NODOUBLES (P);
  G := the graph corresponding to P;
end.
```

THEOREM 2. *DEG generates graphs G of degrees $k_1, \ldots, k_n$ uniformly at random.*

Proof. We show that at each stage in the algorithm, the probability of a pairing P occurring, given that it is in $\mathscr{C}_{l,d}$, is $|\mathscr{C}_{l,d}|^{-1}$. This is true immediately of the initial pairing. It only remains to show that each iteration of the **while** loops in NOLOOPS and NODOUBLES preserves this property.

So assume that P, as in the start of NOLOOPS, is chosen uniformly at random from $\mathscr{C}_{l,d}$. Consider P' obtained by applying a random forward l-switching to P. This particular l-switching is performed with probability $(m_1|\mathscr{C}_{l,d}|)^{-1}$. The probability of accepting this as the new P in NOLOOPS is $m_1 B_2/(m_2 B_1)$, which is at most 1 by Lemma 4(a) and (b). Hence, since there are m_2 l-switchings leading to P', an arbitrary pairing in $\mathscr{C}_{l-1,d}$ occurs as the new P with probability $B_2/(B_1|\mathscr{C}_{l,d}|)$. As this is independent of the old P, and as the only other possible termination within the **while** loop of NOLOOPS is a **restart**, this means that each P in $\mathscr{C}_{l-1,d}$ is equally likely to occur at the beginning of the next iteration of the **while** loop. The analysis of NODOUBLES is similar, and the theorem follows. □

THEOREM 3. *DEG can be implemented so that it generates graphs with n vertices of degrees $1 \le k_1, \ldots, k_n \le k$ uniformly at random in expected time $O(M + M_2^2)$ per graph, provided $k^3 = O(M^2/M_2)$ and $k^3 = o(M + M_2)$.*

Note. $M + M_2^2 \le nk + n^2k^4$. Also, $k = O(M^{1/4})$ implies $k^3 = O(M^2/M_2)$ and $k^3 = o(M)$.

Proof. We bound the expected time, per successful termination of DEG, in a sequence of repeated runs. For this, we take an upper bound on the time taken before a **restart** or successful termination, and then divide by the probability of not restarting in a single run.

Consider firstly the case $k^3 = o(M^2/M_2)$. The initial pairing P can be generated by choosing, first, a mate for one point at random from all the other points, second, a mate for another unused point at random from the remaining points, and so on. Assuming that a random number in $[1, \ldots, j]$, $j \leq M$, can be generated in constant time, the time taken to generate P is $O(M)$. Checking for triple pairs and double loops, and finding the numbers of double pairs and loops can be done in time $O(M)$ with an adjacency matrix (whose initialisation can be avoided by using pointers). By Lemmas 2 and 3′, a **restart** occurs at this point with probability at most $\frac{3}{4} + o(1)$.

It only remains to bound the time taken by a run of NOLOOPS or NODOUBLES. We analyse NODOUBLES in detail, since better bounds can easily be obtained for NOLOOPS, due to the smaller number of loops than double pairs in the worst case.

Note that NODOUBLES is only called if there are at most $(M_2/M)^2$ double pairs, and so this is an upper bound on the number of iterations of the **while** loop. This also means that we can assume $M_2 \geq M$ in NODOUBLES; otherwise, it cannot be called. We can obtain P' easily enough by mimicking the proof of Lemma 4(c): choosing p_2, p_3, p_1, and p_4 at random in one of $4dM^2 = B_3$ ways, with each choice equally likely. At this point, if the d-switching cannot legally be performed, **restart**. The probability of not restarting here is m_3/B_3, so m_3 does not need to be computed. However, the time taken to compute m_4 determines the overall expected run-time. We need m_4 since the next thing to do is to **restart** with probability $1 - B_4/m_4$, thus achieving the desired probability $1 - m_3B_4/(m_4B_3)$ of a **restart** in this step of the algorithm. Note that by Lemma 4, this probability is $O((k^2 + d)/M + (kd + k^3)/M_2) = O(k^3/M_2)$ as $M_2 < kM$. Hence, the probability of not restarting during the whole execution of NODOUBLES is $(1 - O(k^3/M_2))^{(M_2/M)^2} = 1 - O(k^3M_2/M^2) = 1 - o(1)$. A similar calculation for NOLOOPS gives $1 - O(k^3/M) = 1 - o(1)$.

If ease of implementation were of prime importance, one could calculate m_4 in time $O(M_2^2)$ by running through all M_2^2 choices of p_1, p_5, p_3, and p_7 as in the proof of Lemma 4(d), and then testing each (in constant time) for allowability of the reverse d-switching. (Recall that $M_2 \geq M$. An adjacency matrix of the graph of the pairing is useful here.) This can be repeated for each iteration of the **while** loop in NODOUBLES; i.e., at most $(M_2/M)^2$ times. The analogous process for NOLOOPS requires $O(MM_2)$, repeated up to M_2/M times. This would give an overall average-case time complexity for NODOUBLES of $O(M_2^4/M^2) = O(k^2M_2^2) = O(n^2k^6)$. On the other hand, instead of recalculating m_4 for each iteration, one could calculate m_4 in the initial pass and then update its value for the next iteration by calculating the change due to the d-switching. Each of the $O(1)$ pairs involved in a switching requires time $O(kM_2)$ to find its contribution

to m_4 (either positive or negative) as one of the pairs to be switched out in a reverse d-switching, and time $O(k^4)$ as one of the pairs to be switched in. The update must be performed at most $(M_2/M)^2 = O(M_2/k^3)$ times, yielding the time complexity $O(M_2^2)$ for NODOUBLES, since $k \le n \le M \le M_2$ here. As seen before, NOLOOPS requires at most $O(M_2^2)$, and we are done in the case $k^3 = o(M^2/M_2)$.

For $k = O(M^2/M_2)$ with $k \neq o(M^2/M_2)$, triple pairs can become a problem. However, by using inclusion–exclusion (or Bonferroni's inequalities) it is easy to show that the number of triple pairs in such a random pairing has asymptotically a Poisson distribution with mean at least cM^2/M_2 for some $c > 0$. (This is similar to the argument in Wormald [6]. Alternatively, it can also be derived by using switchings as in McKay [2].) Hence, the probability of no triple pairs occurring is $\exp -O(1)$. The only other feature of the proof which changes for $k = O(M^2/M_2)$ is the probability of executing NODOUBLES without a **restart**: again, it is $(1 - O(k^3/M_2))^{(M_2/M)^2} = \exp -O(k^3M_2/M^2)$ (as $k^3 = o(M_2)$), which is $\exp -O(1)$. The number of **restarts** required per successful termination is thus expected to be $O(1)$.

We note finally that if $M_2 < M$, then NOLOOPS and NODOUBLES are never called, and the time reduces to $O(Mk^2)$. □

5. Generation of Random Regular Graphs

By Theorems 2 and 3, DEG generates k-regular graphs on n vertices uniformly at random in expected time $O(n^2k^4)$ provided $k = O(n^{1/3})$, since $M = nk$ and $M_2 = nk(k-1)$. We improve this to the following.

THEOREM 4. *DEG can be implemented so as to generate k-regular graphs on n vertices uniformly at random in expected time $O(nk^3)$ per graph, provided $k = O(n^{1/3})$.*

Proof. From the proof of Theorem 3, we see that the theorem will follow if NOLOOPS and NODOUBLES can each be implemented with a maximum time $O(nk^3)$ between **restarts**. To achieve this, we employ the updating technique and moreover calculate m_4 in a very special way to begin with.

Take a pairing P with d double pairs (and no loops or triple pairs), let F denote the set of pairs in double pairs, and let E denote the edge set of the multigraph of the pairing. Note that m_4 is the number of ordered pairs of 2-paths $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)$ satisfying none of the following crite-

ria, where v_1, v_2, v_3, v_4, v_5, and v_6 denote the cells containing p_1, p_2, p_4, p_5, p_6, and p_8, respectively:

(i) $\{p_1, p_2\} \in F$

(ii) $\{p_3, p_4\} \in F$

(iii) $\{p_5, p_6\} \in F$

(iv) $\{p_7, p_8\} \in F$

(v) $v_1 = v_4$

(vi) $v_1 = v_5$

(vii) $v_1 = v_6$

(viii) $v_2 = v_4$

(ix) $v_2 = v_5$

(x) $v_2 = v_6$

(xi) $v_3 = v_4$

(xii) $v_3 = v_5$

(xiii) $v_3 = v_6$

(xiv) $v_1v_4 \in E$

(xv) $v_2v_5 \in E$

(xvi) $v_3v_6 \in E$.

Specifying the negatives of conditions (i)–(iv) and (xiv)–(xvi) ensures that the switching will not involve double edges, and given that, (v)–(xiii) ensure that $v_1, \ldots, v_6$ are distinct. For $i = 1, \ldots, 16$, let D_i denote the set of ordered pairs of 2-paths satisfying the ith criterion. Then by inclusion–

exclusion,

$$m_4 = \sum_{i=0}^{16} (-1)^i \sum |D_{j_1} \cap \ldots \cap D_{j_i}|, \tag{1}$$

where the second sum is for $1 \le j_1 < \ldots < j_i \le 16$, and the empty summation is taken to be $n^2k^2(k-1)^2$ (the number of ordered pairs of 2-paths with no restrictions).

To compute some of the terms in (1), we use the 3-path structure of the pairing, defined as follows. Let T denote the set of points, let V denote the vertex set of the multigraph, let $a, d \in V$ and let b and c each denote either a point or a special symbol ($*$, say) used to mark a sum over all points. For $b, c \in T$, let $s_{a,b,c,d}$ denote 1 if there is a 3-path beginning with b which is in cell a, and ending with c which is in cell d, and 0 otherwise. Then put $s_{a,*,c,d} = \sum_{p \in T} s_{a,p,c,d}$, and define $s_{a,b,*,d}$ and $s_{a,*,*,d}$ similarly. The *3-path structure* of the pairing is the set of numbers $s_{a,b,c,d}$.

The non-zero elements of the 3-path structure can be determined in time $O(nk^3)$ by running through all 3-paths and incrementing the relevant numbers (which can be stored in an $O(n^3)$ array with constant look-up time). We call this the *s-array*. Records of which entries are non-zero can be stored in an $O(nk^3)$ linked list (so that all such entries can later be found in time $O(nk^3)$), called the *s-list*, with a pointer from each *s*-array element to the corresponding *s*-list element (for constant-time deletion and insertion, and eliminating the need to initialise the *s*-array to zeros). When a *d*-switching is performed, each edge inserted or deleted occurs in $O(k^2)$ 3-paths and hence the above data structure can be updated in time $O(k^2)$.

We claim that each of the terms in the second sum in (1) can be computed in constant time from a set of numbers, including the 3-path structure, which is computable in time $O(nk^3)$ and can be updated after a single *d*-switching in time $O(k^2)$. This gives the stated complexity of $O(nk^3)$.

It only remains to explain how to deal with the 2^{16} terms in (1). We have already treated the case $i = 0$, and each of the 16 terms for $i = 1$ are similar because the answer does not depend on the pairing given n, k, and d. For example, $|D_4| = nk^2(k-1)^2$, since for $v_1 = v_4$ we can choose p_1, p_2, p_3, p_4 in $nk(k-1)$ ways, then choose p_5 to be anything in v_1 (in k ways), and then p_6, p_7, p_8 in $k-1$ ways. Similarly, $|D_{14}| = (nk - 2d)k^2(k-1)^2$, since there are $nk - 2d$ ordered pairs of adjacent vertices (eligible for (v_1, v_4)) and $k(k-1)$ ways to choose each 2-path from there.

We now turn to $i \ge 2$. For these cases, possibly the most difficult being $D_{14} \cap D_{16}$, some computation needs to be done on the given pairing. Consider $D_{14} \cap D_{16}$ first. For this, we wish to find the number of possibilities for pairs $\{p_1, p_2\}, \{p_3, p_4\}, \{p_5, p_6\}, \{p_7, p_8\}$ such that $\{p_2, p_1\}$ and

$\{p_5, p_6\}$ form the beginning and ending of a 3-path, as do $\{p_3, p_4\}$ and $\{p_8, p_7\}$, and p_2 and p_3 share a common cell v_2, p_6 and p_7 share a common cell v_5, $p_2 \neq p_3$ and $p_6 \neq p_7$. Without the last two constraints, this number is $\sum_{v_1, v_2 \in V} s^2_{v_1, *, *, v_2}$ and similarly by inclusion–exclusion,

$$|D_{14} \cap D_{16}| = \sum_{v_1, v_2 \in V} \left(s^2_{v_1, *, *, v_2} + s_{v_1, *, *, v_2} - \sum_{p \in P} \left(s^2_{v_1, p, *, v_2} + s^2_{v_1, *, p, v_2} \right) \right).$$

This can be computed in time $O(nk^3)$, using the s-list to find the non-zero terms. Moreover, the value of each term in this expression can be updated to time $O(k^2)$ when a d-switching is performed since at most this many terms in the s-array are affected.

We only have 65,518 terms in (1) left to deal with. We leave these as an exercise for the reader. We do have the following hints: $D_{14} \cap D_{15} \cap D_{16}$ can be done easily by restricting the calculation for $D_{14} \cap D_{16}$ to the case that v_2 and v_5 are adjacent. Combinations such as $D_5 \cap D_{16}$ are probably best handled using an array of data for 2-paths as well as 3-paths. Anything involving any of D_1 to D_4 is as easy as the corresponding case excluding these sets, since there are $O(k^2)$ double pairs, which can be located in advance in time $O(nk^2)$ and listed for ready reference in the future. Often, it will suffice to look at all such double edges, and all pairs of 2-paths emanating from either end (in time $O(k^6) = O(nk^3)$). An adjacency matrix for the multigraph may come in handy. (Again, pointers can be used to eliminate the need to initialise this.) From here, combinations involving more of the D_i get progressively easier to deal with, and almost all of the terms in (1) can be shown to be 0 always. □

6. Bipartite Graphs

Minor modifications and/or simplifications to the methods presented in the previous sections are sufficient to generate random bipartite graphs with vertices in the first part, V_1 say, having degrees $k_{1,1}, \ldots, k_{n_1,1}$ and those in the second part, V_2 say, having degrees $k_{1,2}, \ldots, k_{n_2,2}$.

Our model of graphs is now slightly different: we consider $2M$ points, with M of them arranged in cells of sizes $k_{1,1}, \ldots, k_{n_1,1}$ (corresponding to vertices in V_1) and the other M in cells of sizes $k_{1,2}, \ldots, k_{n_2,2}$ (corresponding to vertices in V_2). A pairing P is still a pairing of the $2M$ points, but is now restricted so that each pair contains one point in a cell in V_1 and one point in a cell in V_2. Hence, loops are impossible, and so only d-switchings are required.

Define $M_{2,j} = \sum_{i=1}^{n_j} k_{i,j}(k_{i,j} - 1)$ for $j = 1$ and 2, and $M_2 = M_{2,1} + M_{2,2}$. The arguments used to establish Lemmas 1, 2, and 3′ can be used with

minor alterations to establish the following results. We assume $k_{i,j} \leq k$ for all i, and $j = 1, 2$.

LEMMA 1B. *The probability of t given pairs occurring in P is at most $(M - t)^{-t}$.*

LEMMA 2B. *The probability that P contains at least one triple pair is*

$$O\left(k^2 M_{2,1} M_{2,2} / M^3\right).$$

LEMMA 3B′. *We have*

$$\Pr\left\{d > M_{2,1} M_{2,2} / M^2\right\} < \tfrac{1}{2} + o(1).$$

We also need to modify the definition of a d-switching, to the effect that the points p_2, p_3, p_5, and p_8 must be in V_1 whilst p_1, p_4, p_6, and p_7 must be in V_2. Then in place of Lemma 4, we have the following, with $\mathscr{C}_{0,d}$ defined as for graphs.

LEMMA 4B. *For each of the following operations, the number, m, of ways of applying the operation is*

(a) *forward d-switching $\mathscr{C}_{0,d} \to \mathscr{C}_{0,d-1}$:*

$$2dM^2 \geq m \geq 2dM^2\left(1 - O\left(\frac{k^2 + d}{M}\right)\right);$$

(b) *backward d-switching $\mathscr{C}_{0,d-1} \to \mathscr{C}_{0,d}$:*

$$M_{2,1} M_{2,2} \geq m \geq M_{2,1} M_{2,2} - (k-1) M_2\left(4d + 2k + 3(k-1)^2/2\right).$$

In order to generate random bipartite graphs, we therefore use the following algorithm. First, NODOUBLES will be as described previously, but with the new definition of pairings and d-switchings, and with B_3 and B_4 denoting the upper bound in Lemma 4B (a) and the lower bound in Lemma 4B (b), respectively. We assume $M \geq n_1$ and $M \geq n_2$.

```
procedure BIDEG (n_1, k_{1,1}, ..., k_{n_1,1}, n_2, k_{1,2}, ..., k_{n_2,2});
begin
  select a pairing uniformly at random from the pairings of degrees k_{1,1}, ..., k_{n_1,1}
  in V_1 and k_{1,2}, ..., k_{n_2,2} in V_2;
  if P has any multiple pairs of cardinality greater than 2,
      or more than M_{2,1}M_{2,2}/M double pairs then restart;
  NODOUBLES (P);
  G := the (bipartite) graph corresponding to P;
end.
```

We now have

THEOREM 2B. *BIDEG generates bipartite graphs G of degrees $k_{1,1}, \ldots, k_{n_1,1}$ in the first part and degrees $k_{1,2}, \ldots, k_{n_2,2}$ in the second part uniformly at random.*

An analogue of Theorem 3 also follows by similar arguments. In particular, if $k = O(M^{1/4})$ then the expected time taken by BIDEG is $O(n^2k^4)$ per graph, where $n = n_1 + n_2$. This improves to $O(nk^3)$ for k-regular bipartite graphs, using the method of implementation discussed in the proof of Theorem 4.

REFERENCES

1. B. BOLLOBÁS, "Random Graphs," Academic Press, London, 1985.
2. B. D. MCKAY, Asymptotics for symmetric 0-1 matrices with prescribed row sums, *Ars Combin. A***19** (1985), 15–25.
3. B. D. MCKAY AND N. C. WORMALD, Asymptotic enumeration by degree sequence of graphs with degree $o(n^{1/2})$, to appear.
4. A. SINCLAIR AND M. JERRUM, "Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chairs," Internal report CSR-241-87, Department of Computer Science, University of Edinburgh, 1987.
5. N. C. WORMALD, Generating random regular graphs, *J. Algorithms* **5** (1984), 247–280.
6. N. C. WORMALD, The asymptotic distribution of short cycles in random regular graphs, *J. Combin. Theory Ser. B* **31** (1981), 156–167.