Combinatorics, Probability and Computing (2006) 15, 333–343. © 2006 Cambridge University Press
doi:10.1017/S0963548305007303 Printed in the United Kingdom

Generating Outerplanar Graphs Uniformly at Random

MANUEL BODIRSKY and MIHYUN KANG[†]

Humboldt-Universität zu Berlin, Germany
(e-mail: {bodirsky,kang}@informatik.hu-berlin.de)

Received 28 April 2003; revised 5 March 2004

We show how to generate labelled and unlabelled outerplanar graphs with n vertices uniformly at random in polynomial time in n. To generate labelled outerplanar graphs, we present a counting technique using the decomposition of a graph according to its block structure, and compute the exact number of labelled outerplanar graphs. This allows us to make the correct probabilistic choices in a recursive generation of uniformly distributed outerplanar graphs.

Next we modify our formulas to also count *rooted unlabelled* graphs, and finally show how to use these formulas in a Las Vegas algorithm to generate unlabelled outerplanar graphs uniformly at random in expected polynomial time.

1. Introduction

There are several fields of applications of efficient algorithms that generate random combinatorial structures. We can use such a generation procedure as an experimental tool to verify properties of these structures that hold almost always. It can also be used to produce test instances for other algorithms on these structures. We can then measure the average running time of these algorithms on random instances.

Outerplanar graphs are planar graphs with an embedding where every vertex lies on the outer face. They are interesting examples of combinatorial structures with respect to the mentioned motivations of random generation procedures, having numerous applications in various areas of computer and natural sciences. Many computational problems that are hard in general cases become tractable for outerplanar graphs [21, 26]. But still, their structure is rich enough that many computational tasks remain challenging when the input is restricted to outerplanar graphs [6, 15]. Outerplanar graphs are also attracting increasing interest in graph drawing [2, 5, 19].

[†] Members of the European Graduate Program 'Combinatorics, Geometry, and Computation', supported by the DFG (GRK 588/1).

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

Often one is interested not in the worst-case running time of an algorithm, but in the running time on a *typical* instance. The so-called *average-case complexity* of a problem then depends on a probability distribution on the problem instances. In this paper, we will focus on the generation of random outerplanar graphs according to their *uniform distribution*. Algorithms that generate combinatorial structures uniformly at random can thus be used to evaluate the average-case running time of the algorithm.

The choice of the uniform distribution might not be appropriate to study the average-case behaviour of an algorithm for a certain application. But if the distribution of a *typical* input instance in an application is not known, the uniform distribution is a well-understood and reasonable choice from which to draw test instances. The uniform distribution also plays an important role in the theory of average-case complexity classes (see [16, 20]).

Besides outerplanar graphs, we would like to mention a related random structure that has recently attracted attention: unlike the random graph, little is yet known about the *random planar graph*. Random planar *maps*, *i.e.*, embedded planar graphs, were studied in [3], and it is known how to generate rooted 3-connected planar maps uniformly at random [28]. Osthus, Prömel and Taraz [27], and Bender, Gao and Wormald [4] derived upper and lower bounds on the asymptotic number of planar graphs, and Gerke and McDiarmid [14] investigated the expected number of edges in a random planar graph. Denise, Vasconcellos and Welsh [9] specified a Markov chain on the space of all planar graphs, whose limit distribution is the uniform distribution over all planar graphs.

Markov chains that uniformly generate some combinatorial objects have turned out to be a powerful tool for the theory of random structures, since they can help to prove properties of a random instance of the combinatorial object. Markov chains with a uniform stationary distribution can be used as *Monte Carlo* algorithms for the generation of random structures that are approximately uniformly distributed. These Monte Carlo algorithms are called *efficient* if the Markov chain is *rapidly mixing*, that is, the number of steps until the Markov chain is close to the uniform distribution (which is called the *mixing time*) is polynomial.

The Markov chain technique leads to efficient algorithms for the generation of random trees and triangulations of convex polygons [22, 24]. On the other hand, it is not clear whether the Markov chain in [9] for the uniform generation of planar graphs is rapidly mixing. Interestingly, the difficulties in proving any nontrivial upper bound on the mixing time already appear for Markov chains that generate *outerplanar graphs* uniformly at random.

As in the case of planar graphs, outerplanar graphs can be characterized in terms of forbidden minors (namely $K_{2,3}$ and K_4, [8]). Mitchell [23] gave a linear time algorithm to recognize outerplanar and maximal outerplanar graphs. Maximal outerplanar graphs have a simple structure. They can be seen as a triangulation of a convex polygon. Their dual graph forms a binary tree structure and thus they can be counted by the Catalan numbers, and can be efficiently generated [11, 13]. Biconnected outerplanar graphs can be seen as dissections of a convex polygon, and their number can be counted by the number of the Schröder number [29, 30], also called the bracketing number. There are no such simple formulas known for general outerplanar graphs.

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

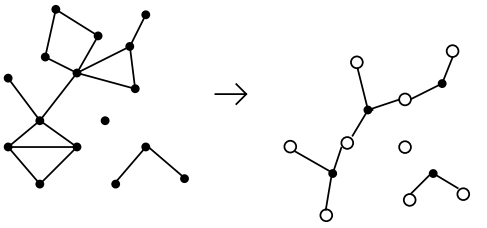


Figure 1. The block structure of a graph.

In this paper we show how to efficiently generate random labelled outerplanar graphs. This means, for a given set of n vertices, we will generate a random labelled outerplanar graph in polynomial time in n. Every labelled outerplanar graph will be generated with equal probability, given that our algorithm has access to $O(n^2 \log n)$ random bits. We will then show how to modify the algorithm to generate unlabelled connected outerplanar graphs uniformly at random in expected polynomial time.

Both for the labelled and the unlabelled case we first derive counting formulas, which can be evaluated efficiently with dynamic programming. To count and generate the graphs we make use of the *block structure* of the graph, which is essentially the tree structure formed by the biconnected components and cutvertices of the graph. The generation procedure can be seen as the reversed decomposition of a graph along its block structure, and is guided by the probabilities computed with the counting formulas.

2. Outerplanar graphs

A graph G is *outerplanar* if there is an embedding of G in the plane that has a face containing all the vertices of G. We will always draw this face as the outer face. To describe how outerplanar graphs look, the well-known graph-theoretic concept of the block structure of a graph (see, *e.g.*, Figure 1) is useful.

We will first recall and fix some graph-theoretic notions (using the conventions in [12]). A maximal connected subgraph of a graph G is called a *component* of G. A *cutvertex* of a component is a vertex such that the component without this vertex is not connected. A *block* of G is a maximal connected subgraph without a cutvertex. A graph G is called *biconnected* if $|G| > 2$ and $G - \{u\}$ is connected for every vertex $u \in G$. Thus, every block of a graph G is either a maximal biconnected subgraph, an edge (including its ends), or an isolated vertex.

Definition. The *block structure* of a graph G is a forest on two types of vertices: the blocks and the cutvertices of G. A block and a cutvertex are adjacent in the forest if the block contains the vertex.

For convenience, we also call a vertex that is not a cutvertex *adjacent* to a block, if the block contains the vertex. The reason why the decomposition into biconnected components of a graph is so useful is that biconnected outerplanar graphs have a simple structure.

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

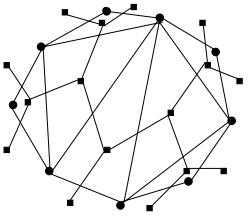


Figure 2. A biconnected outerplanar graph and a tree representation of its dual.

Proposition 2.1. *A biconnected outerplanar graph contains a unique Hamiltonian cycle.*

Without loss of generality we can assume that the vertices of the Hamiltonian cycle of a biconnected outerplanar graph with n vertices are labelled in clockwise order, and form a convex n-gon K. A biconnected outerplanar graph then corresponds to a dissection of K, and its dual graph is again a tree-structure. The number of dissections of K equals the Schröder number $s(n-1)$ (also called the *bracketing number* [29, 30]), which satisfies the following recursion: $s(1) = s(2) = 1$,

$$s(n) = ((6n-9)s(n-1) - (n-3)s(n-2))/n. \tag{2.1}$$

We remark that it is possible to add a parameter m to all recursive counting formulas to specify the number of edges of the corresponding graphs.

3. Generating forests uniformly at random

In this section we want to explain how recursive counting formulas are used to generate random structures uniformly at random. We illustrate the idea with a counting formula for *labelled forests* and *labelled trees*, and obtain a random generator that runs in time $\tilde{O}(n^4)$, where $\tilde{O}$ denotes the asymptotics up to logarithmic factors. We will formulate this simple algorithm here only for explanatory reasons. Note that for trees linear time generators are known [1].

Let $f(n)$ denote the number of labelled forests with n vertices $\{0, 1, \ldots, n-1\}$, and $f_c(n)$ the number of forests with c connected components. Clearly, for $n \geqslant 1$,

$$f(n) = \sum_{c=1}^{n} f_c(n). \tag{3.1}$$

A labelled forest with more than one component has a unique decomposition into the tree that contains the vertex labelled with zero (*i.e.*, the vertex 0), and the forest with the remaining components. Thus

$$f_c(n) = \sum_{i=1}^{n-c+1} \binom{n-1}{i-1} f_1(i) f_{c-1}(n-i). \tag{3.2}$$

To count trees, *i.e.*, connected forests, let $t(n)$ be the number of labelled trees with n vertices, and $t_d(n)$ the number of trees where the vertex 0 has degree d. Clearly $t(n) = \sum_{d=1}^{n-1} t_d(n)$, for $n \geqslant 2$.

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

Generate(n): returns random tree with vertices $0,\ldots,n-1$.
  if $n \leqslant 1$ **then return** $\emptyset$.
  choose the degree d of 0 with probability $t_d(n)/t(n)$.
  return Generate(n,d)

Generate(n,d): returns tree with vertices $0,\ldots,n-1$ where 0 has degree d
  if $d=1$ **then let** $T=$ Generate$(n-1)$; relabel vertex j in T to $j+1$.
  return $T \cup \{0,1\}$
  choose the size i of the split subtree with probability $\binom{n-2}{i-2} t(i) t_{d-1}(n-i)/t_d(n)$.
  choose random subset $\{w_0,\ldots,w_{i-1}\} \subseteq \{1,\ldots,n-1\}$
  $T_1 =$ **Generate**(i); relabel vertex j in T_1 to w_j.
  $T_2 =$ **Generate**$(n-i, d-1)$.
  return $T_1 \cup T_2 \cup \{0, w_0\}$.

Figure 3. Generating labelled trees uniformly at random.

If T is a tree where the vertex 0 has degree one, then we can decompose T into the vertex 0 and the remaining subtree with $n-1$ vertices, hence $t_1(n) = t(n-1)$. If the vertex 0 has degree $d \geqslant 2$, we decompose T into two subtrees. One is the subtree of the vertex 0 containing the vertex 1, and is called the *split subtree*. The other one is the rest of the tree together with the vertex 0. The uniqueness of this decomposition gives rise to a random generation procedure, and correspondingly to a counting formula for the set of labelled trees: for $d \geqslant 2$, $n \geqslant 2$,

$$t_d(n) = \sum_{i=1}^{n-d} \binom{n-2}{i-2} t(i) t_{d-1}(n-i). \tag{3.3}$$

To compute the numbers $t(n)$ and $t_d(n)$ efficiently, we store all of their values in a table to avoid recomputation, which is a technique called *dynamic programming*. Since the number of labelled trees on n vertices is n^{n-2}, the entries of the table must store $O(n \log n)$ many bits. Assuming $O(n \log n \log\log n)$ multiplication algorithm (see, *e.g.*, [7]), the number of computation steps needed to fill the quadratic size table is $O(n^4 \log n \log\log n)$.

As we see in Figure 3, it is an easy exercise to derive a generation procedure, once the exact recursive counting formulas with appropriate parameters are known. This method is often called the *recursive method* initiated by [25]. In the following sections, when dealing with outerplanar graphs, we will therefore concentrate on the presentation of the decomposition strategy and the counting formulas.

4. Decomposing labelled outerplanar graphs

Since the block structure of an outerplanar graph is a forest, we want to count and generate outerplanar graphs similarly as demonstrated in the last section. As we saw there, we can restrict our attention to connected graphs.

In order to decompose an outerplanar graph, we consider two cases: the vertex labelled with the smallest label is either a cutvertex and hence it is contained in more than one

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

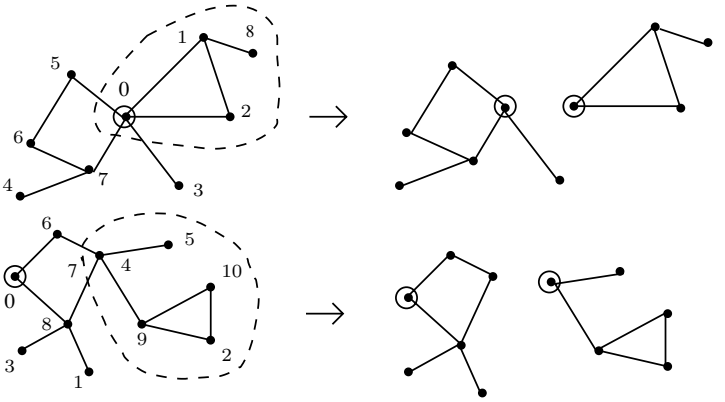


Figure 4. Pulling off the petals from the flower.

block, or it is not a cutvertex and hence it is contained in a unique block. Depending on these two cases, we apply a degree-reduction strategy as in the case of labelled trees: see Figure 4.

Let $p(n)$ be the number of all labelled outerplanar graphs with n vertices $\{0, 1, \ldots, n-1\}$, and $p_d(n)$ the number of all labelled outerplanar graphs with n vertices, where the vertex 0 is adjacent to d blocks. Then, for $n \geqslant 2$, $p(n) = \sum_{d=1}^{n-1} p_d(n)$. Note that $\sum_{d=2}^{n-1} p_d(n)$ counts all outerplanar graphs with n vertices where the vertex 0 is a cutvertex, and $p_1(n)$ all outerplanar graphs with n vertices where the vertex 0 is a non-cutvertex.

Let G be an outerplanar graph with n vertices where the vertex 0 is a cutvertex and is adjacent to d blocks, $d \geqslant 2$. At the vertex 0 we split off the connected component containing the vertex 1 from G. Then in the remaining graph the vertex 0 is adjacent to $d-1$ blocks. If the split subgraph has i vertices, then there are $\binom{n-2}{i-2}$ ways to choose a vertex set of the split subgraph since two vertices 0, 1 are already contained in the split subgraph. It follows that for $d \geqslant 2$, $n \geqslant 3$,

$$p_d(n) = \sum_{i=2}^{n-d+1} \binom{n-2}{i-2} p_1(i) p_{d-1}(n-i+1).$$

We consider the case that the vertex 0 is a non-cutvertex and hence it is contained in a unique block, called the *root block*. Let $q_c(n)$ be the number of all outerplanar graphs with n vertices, where the smallest c vertices of the root block are not cutvertices. Then clearly $p_1(n) = q_1(n)$.

From such a graph we split off a subgraph attached at the $(c+1)$th-smallest vertex, which might be any kind of outerplanar graph. Then in the remaining graph the $(c+1)$th-smallest vertex of the root block is not a cutvertex. Thus, for $c \geqslant 1$, $n \geqslant 3$,

$$q_c(n) = \sum_{i=1}^{n-b+1} \binom{n-1}{i-1} p(i) q_{c+1}(n-i+1).$$

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

If the smallest n vertices of the root block of an outerplanar graph with n vertices are not cutvertices, the graph is biconnected. As explained in Section 2,

$$q_n(n) = \frac{(n-1)!}{2} s(n-1) \tag{4.1}$$

where $s(n)$ is the nth Schröder number.

Thus we obtain a complete set of recursive formulas that count labelled outerplanar graphs. As mentioned and illustrated in the remarks in Section 3 the decomposition and counting formulas of labelled outerplanar graphs give rise to an efficient uniform random generation procedure. For the generation of biconnected outerplanar graphs we use the tree structure of its dual as mentioned in Section 2. We want to remark that the decomposition and counting technique is not limited to outerplanar graphs, but works for all graph classes, and it makes sense if the number of biconnected graphs can be computed.

Figure 5 shows some results from an implementation of the counting formulas. The expected edge density of a random labelled outerplanar graph is known to be $\geqslant 7/5$ [14]. The values for up to 100 vertices, depending on the connectivity, are shown in Figure 5(a). The parameter m in Figures 5(b), (c) denotes the number of edges. Figure 5(b) shows the number of c-connected outerplanar graphs, and Figures 5(c), (d) the expected connectivity.

5. Unlabelled outerplanar graphs

Counting unlabelled structures, *i.e.*, counting isomorphism classes, is usually much harder than counting labelled structures. It is, however, possible to derive closed formulas for some classes of graphs, for instance for unlabelled connected, unlabelled bipartite, or unlabelled Eulerian graphs (see, *e.g.*, [17]).

It turns out that the exact number of unlabelled but *rooted connected* outerplanar graphs can be found by the same decomposition strategy and modifications of the presented counting methods. Throughout this section we assume that the outerplanar graphs are unlabelled and connected. A *rooted* outerplanar graph is an outerplanar graph with a special designated vertex, called the *root*. We count rooted outerplanar graphs up to isomorphisms that fix the root vertex. Thus, isomorphic outerplanar graphs that only differ in the position of the root are counted several times.

We present a uniform generation procedure for outerplanar graphs using *rejection sampling*. This leads to a Las Vegas algorithm, *i.e.*, an algorithm that produces every outerplanar graph precisely with the same probability, but has a running time that depends on the random bits used by the computer, and is polynomial in expectation. The algorithm first computes a random *rooted* outerplanar graph. Then it outputs the graph with probability $\frac{1}{o}$, where o is the size of the orbit of the root in the automorphism group of the unrooted outerplanar graph. Otherwise, the algorithm is restarted. Since the probability that the graph is rejected is at most $1 - \frac{1}{n}$, the expected number of restarts is at most

$$\sum_{i=1}^{\infty} i \left(1 - \frac{1}{n}\right)^{i-1} \frac{1}{n} = n.$$

It is easy to compute the orbit of the root using an efficient linear time algorithm for isomorphism testing of outerplanar graphs (one could even use the linear time isomorphism

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

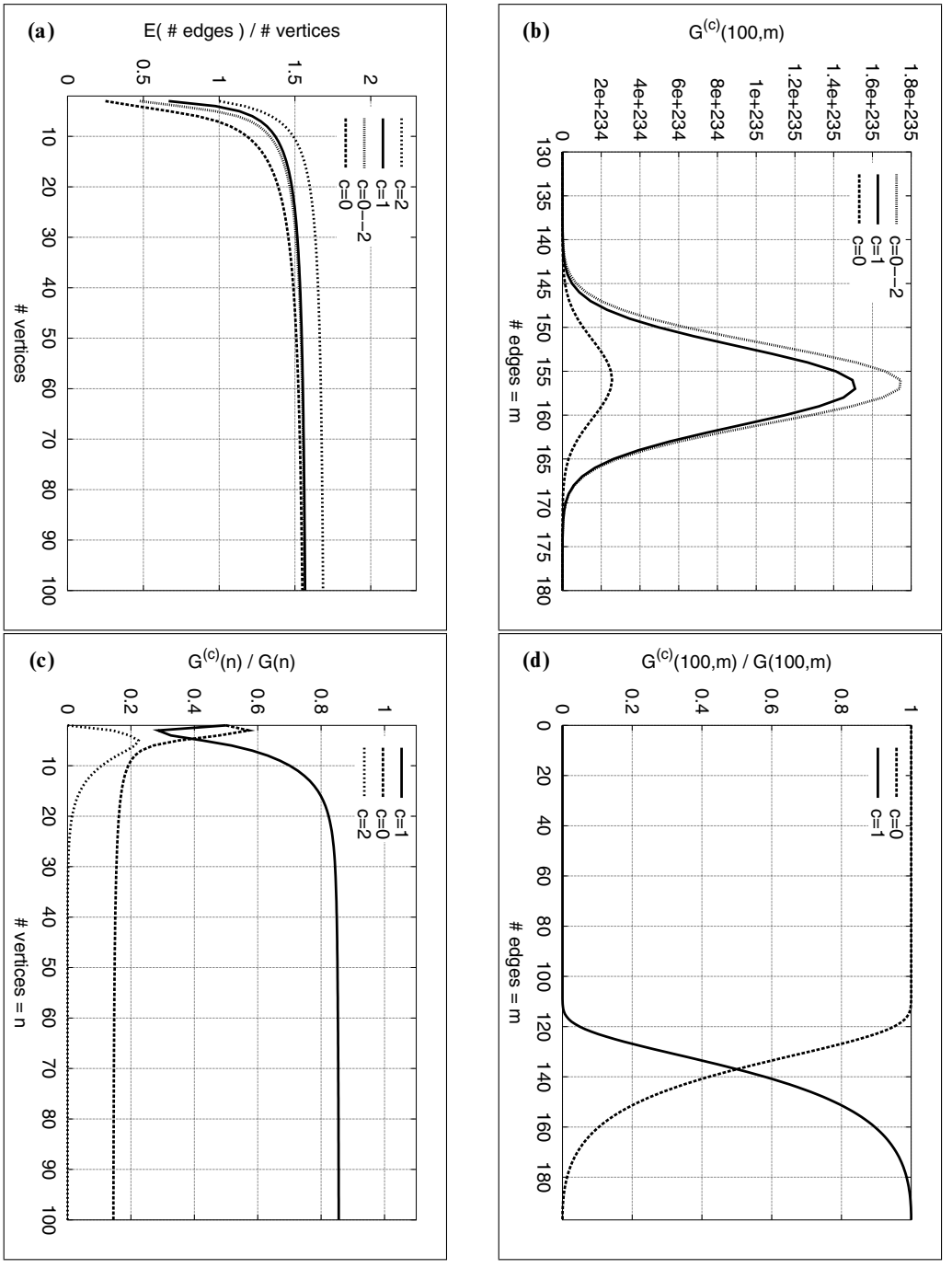


Figure 5. Some counting results for labelled outerplanar graphs on 100 vertices. (a) Edge density of a random labelled outerplanar graph. (b) Number of c-connected labelled outerplanar graphs. (c), (d) Expected connectivity.

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

testing algorithm for planar graphs [18], testing for each position of the root in the graph whether there is an automorphism that maps the vertex of the root to this vertex).

Let $p(n)$ be the number of all rooted (unlabelled connected) outerplanar graphs with n vertices. Again, we distinguish two cases: the case where the root is a cutvertex, and the case with a unique root block. In the first case, the graph without the root has several connected components. Let $p_l(n)$ be the number of all outerplanar graphs with n vertices where all these components have size $\leqslant l$; thus clearly $p_1(n) = 1$ (which counts a so-called *star* graph). Let $q(n)$ be the number of all outerplanar graphs with n vertices and a unique root block. Then, summing over the number k of components of size l, we get for $n \geqslant 2$ and $1 \leqslant l \leqslant n-1$,

$$p(n) = p_{n-1}(n), \qquad p_l(n) = \sum_{k=0}^{\lfloor (n-1)/l \rfloor} \binom{q(l+1)+k-1}{k} p_{l-1}(n-kl).$$

In the second case, where the outerplanar graph has a unique root block, it is not so easy to avoid double counting. The number of outerplanar graphs where the root block consists of a single edge is $p(n-1)$. But otherwise we have to distinguish between two cases: The outerplanar graph might or might not have a symmetry; in other words, it might or might not have a nontrivial automorphism that fixes the root vertex, and exchanges the two vertices on the unique Hamiltonian cycle of the root block that have the same distance to the root. According to that, let $a(n)$ denote the number of *asymmetric* outerplanar graphs with n vertices. Correspondingly $b(n)$ denotes the number of *symmetric* such graphs. Then for $n \geqslant 3$,

$$q(n) = p(n-1) + a(n) + b(n).$$

To compute the symmetric (resp. asymmetric) outerplanar graphs with a unique root block, let $b_l(n)$ (resp. $a_l(n)$) be the number of symmetric (resp. asymmetric) outerplanar graphs with n vertices that do not have a cutvertex at distance $\leqslant l$ from the root on the unique Hamiltonian cycle of the root block. For $n \geqslant 3$,

$$b_l(n) = \sum_{i=1} p(i+1) b_{l+1}(n-2i).$$

An outerplanar graph with a unique root block can be asymmetric for three disjoint reasons: either the graph formed by deleting a graph attached to the cut vertex with distance l to the root is already asymmetric, or two graphs of different size are attached to the cutvertices with distance l to the root, or the two graphs attached to the cutvertices with distance l to the root are distinct for another reason. Then

$$a_l(n) = \sum_{i,j=0} p(i+1)p(j+1)a_{l+1}(n-i-j) + \sum_{i \neq j \geqslant 0} p(i+1)p(j+1)b_{l+1}(n-i-j) + \sum_{i=0} \binom{p(i+1)}{2} b_{l+1}(n-2i).$$

It remains to clarify the initial cases for $a_l(n)$ and $b_l(n)$, where $l = n$. In the sense above, we also distinguish between symmetric and asymmetric (with respect to the root vertex) *biconnected* outerplanar graphs. So, let $c(n)$ be the number of symmetric biconnected

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

Table 1. Usage of computational resources. Growth of the *expected* running time in the case of unlabelled generation. $\tilde{O}(\cdot)$ denotes growth up to logarithmic factors.

	precomputation time	space	generation time and random bits
labelled	$\tilde{O}(n^4)$	$O(n^3 \log n)$	$\tilde{O}(n^2)$
unlabelled	$\tilde{O}(n^4)$	$O(n^3)$	$\tilde{O}(n^3)$

graphs with n vertices. Clearly $b_n(n) = c(n)$ and $a_n(n) = (s(n-1) - c(n))/2$, where $s(n)$ is the nth Schröder number that was already mentioned in Section 2. Using $s(n)$ we can also calculate $c(n)$ as follows: $c(1) = c(2) = c(3) = 1$ and for $n \geqslant 2$, $0 \leqslant l \leqslant \lceil n/2 \rceil$,

$$c(2n) = c(2n+1) = c(2n-1) + 2\sum_{i=2}^{n} s(i)c(2n+1-2i).$$

Thus, we completed the exact counting formulas for rooted outerplanar graphs. It is easy to see that the number of outerplanar graphs only grows exponentially, and thus the values of this section can be stored in linear space.

6. Conclusion

Brute-force algorithms to generate random outerplanar graphs uniformly at random require exponential time, and Markov chain Monte Carlo methods have unknown mixing times and only approximate the uniform distribution. We have developed a flexible polynomial time algorithm, which can be adapted to generate and count labelled outerplanar graphs, labelled connected outerplanar graphs, unlabelled connected outerplanar graphs, or unlabelled symmetric rooted outerplanar graphs, for example, uniformly at random.

To increase the efficiency of the algorithm (see Table 1) one can use floating point numbers instead of arbitrary precision integer arithmetic. Since the algorithm can base its decisions on $O(\log n)$ bits in most cases, it might even be possible to get an exact uniform generator if we use certified floating-point arithmetics: see, *e.g.*, [10].

Acknowledgements

We would like to thank Mike Löffler for an implementation of the counting formulas and the preparation of the diagrams.

References

[1] Alonso, L., Remy, J. and Schott, R. (1997) A linear-time algorithm for the generation of trees. *Algorithmica* **17** 162–182.
[2] Atallah, M. J. and Manning, J. (1992) Fast detection and display of symmetry in outerplanar graphs. *Discrete Appl. Math.* **39** 13–35.
[3] Banderier, C., Flajolet, P., Schaeffer, G. and Soria, M. (2001) Random maps, coalescing saddles, singularity analysis, and Airy phenomena. *Random Struct. Alg.* **19** 194–246.

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press

[4] Bender, A., Gao, Z. and Wormald, N. (2002) The number of labeled 2-connected planar graphs. *Electron. J. Combin* **9** #43.
[5] Biedl, T. (2002) Drawing outer-planar graphs in o(n log n) area. In *10th Annual Conference on Graph Drawing*.
[6] Bodlaender, H. L. and Fomin, F. V. (2001) Approximation of pathwidth of outerplanar graphs. In *Proc. 27th International Workshop on Graph-Theoretic Concepts in Computer Science*, Vol. 2204 of *Lecture Notes in Computer Science*, pp. 166–176.
[7] Bürgisser, P., Clausen, M. and Shokrollahi, M. (1997) *Algebraic Complexity Theory*, Vol. 315 of *Grundlehren der Mathematischen Wissenschaften*, Springer.
[8] Chartrand, G. and Harary, F. (1967) Planar permutation graphs. *Ann. Inst. H. Poincaré, Sec. B3*, pp. 433–438.
[9] Denise, A., Vasconcellos, M. and Welsh, D. (1996) The random planar graph. *Congressus Numerantium* **113** 61–79.
[10] Denise, A. and Zimmermann, P. (1999) Uniform random generation of decomposable structures using floating-point arithmetic. *Theoret. Comput. Sci.* **218** 233–248.
[11] Devroye, L., Flajolet, P., Hurtado, F., Noy, M. and Steiger, W. (1999) Properties of random triangulations and trees. *Discrete Comput. Geom.* **22** 105–117.
[12] Diestel, R. (1997) *Graph Theory*, Springer, New York.
[13] Epstein, P. and Sack, J.-R. (1994) Generating triangulations at random. *ACM Trans. Model. Comput. Simul.* **4** 267–278.
[14] Gerke, S. and McDiarmid, C. On the number of edges in random planar graphs. Submitted.
[15] Govindran, R., Langston, M. and Yan, X. (1998) Approximating the pathwidth of outerplanar graphs. *Inform. Process. Lett.* **68** 17–23.
[16] Gurevich, Y. (1991) Average case completeness. *J. Comput. System Sci.* **42** 346–398.
[17] Harary, R. and Palmer, E. M. (1973) *Graphical Enumeration*, Academic Press, New York and London.
[18] Hopcroft, J. and Tarjan, R. (1971) A v^2 algorithm for determining isomorphism of planar graphs. *Inform. Process. Lett.* **1**(1) 32–34.
[19] Lenhart, W. and Liotta, G. (1997) Proximity drawings of outerplanar graphs. In *Graph Drawing; Proc. GD '96* (S. North, ed.), Vol. 1190 of *Lecture Notes in Computer Science*, pp. 286–302.
[20] Levin, L. (1986) Average case complete problems. *SIAM J. Comput.* **15** 285–286.
[21] Leydold, J. and Stadler, P. F. (1998) Minimal cycle bases of outerplanar graphs. *Electron J. Combin.* **5** #16.
[22] McShine, L. and Tetali, P. (1999) On the mixing time of the triangulation walk and other Catalan structures. In Vol. 43 of *DIMACS Series in Discrete Math. Theor. Comput. Sci.*, pp. 147–159.
[23] Mitchell, S. L. (1979) Linear algorithms to recognize outerplanar and maximal outerplanar graphs. *Inform. Process. Lett.* **9** 229–232.
[24] Molloy, M., Reed, B. and Steiger, W. (1999) On the mixing rate of the triangulation walk. In Vol. 43 of *DIMACS Series in Discrete Math. Theor. Comput. Sci.*, pp. 179–190.
[25] Nijenhuis, A. and Wilf, H. (1979) *Combinatorial Algorithms*, Academic Press.
[26] Okamura, H. and Seymour, P. (1981) Multicommodity flows in planar graphs. *J. Combin. Theory Ser. B* **31** 75–81.
[27] Osthus, D., Prömel, H. J. and Taraz, A. On random planar graphs, the number of planar graphs and their triangulations. *J. Combin. Theory Ser. B*, to appear.
[28] Schaeffer, G. (1999) Random sampling of large planar maps and convex polyhedra. In *Proc. 31st Annual ACM Symposium on Theory of Computing* (STOC'99, Atlanta, Georgia, May 1999), ACM Press, pp. 760–769.
[29] Schröder, E. (1870) Vier combinatorische Probleme. *Z. Math. Physik* **15** 361–376.
[30] Sloane, N. J. A. Sequence A001003: The on-line encyclopedia of integer sequences, `http://www.research.att.com/~njas/sequences/`.

https://doi.org/10.1017/S0963548305007303 Published online by Cambridge University Press