Computational Pólya theory*

Mark Jerrum†
Department of Computer Science
University of Edinburgh
United Kingdom

Abstract

A permutation group G of degree n has a natural induced action on words of length n over a finite alphabet Σ, in which the image x^g of x under permutation $g \in G$ is obtained by permuting the positions of symbols in x according to g. The key result in "Pólya theory" is that the number of orbits of this action is given by an evaluation of the *cycle-index polynomial* $P_G(z_1, \ldots, z_n)$ of G at the point $z_1 = \cdots = z_n = |\Sigma|$. In many cases it is possible to count the number of essentially distinct instances of a combinatorial structure of a given size by evaluating the cycle-index polynomial of an appropriate symmetry group G.

We address the question "to what extent can Pólya theory be mechanised?" There are compelling complexity-theoretic reasons for believing that there is no efficient, uniform procedure for computing the cycle-index polynomial exactly, but less is known about approximate evaluation, say to within a specified relative error. The known results — positive and negative — will be surveyed.

1 Preliminaries

This article is concerned with a topic in computational algebra, which combines aspects of combinatorics, algorithmics, and computational complexity. On the assumption that most readers will be unfamiliar with at least one of these, the first section aims to give a brief account of key facts.

*To be presented at the 15th British Combinatorial Conference, Stirling, July 1995.

†The author is supported in part by grant GR/F 90363 of the UK Science and Engineering Research Council, and by Esprit Working Group No. 7097, "RAND." Address for correspondence: Department of Computer Science, University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ, United Kingdom.

1.1 Elementary group-theoretic preliminaries

Let Σ be a finite alphabet of cardinality k, and G a permutation group on $[n] = \{0, \ldots, n-1\}$. The group G has a natural induced action on the set Σ^n of all words of length n over the alphabet Σ, under which the word $\alpha = x_0 x_1 \ldots x_{n-1}$ is mapped by the permutation $g \in G$ to the word $\alpha^g = y_0 y_1 \ldots y_{n-1}$ defined by $y_j = x_i$ for all $i, j \in [n]$ satisfying $i^g = j$.[1] The action of G partitions Σ^n into a number of *orbits*, these being the equivalence classes of Σ^n under the equivalence relation that identifies α and β whenever there exists $g \in G$ mapping α to β.

By way of example, let $\Sigma = \{0, 1\}$, and $n = m^2$ for some integer m. Interpret each element of Σ^n as the adjacency matrix of an m-vertex directed graph Γ. Let G be the permutation group of degree n and order $m!$ whose elements correspond to permutations of the vertex set of Γ, i.e., to simultaneous permutations of the rows and columns of the adjacency matrix. Then the orbits of Σ^n under the action of G correspond naturally to unlabelled directed graphs on m vertices. Many other sets of unlabelled combinatorial structures can be obtained using a like construction.

What has just been described is the setting for *Pólya theory*, the key result of which is an expression for the number of orbits in terms of the *cycle-index polynomial* of G [4]. This polynomial, in the variables $z_1, z_2, \ldots, z_n$, is defined to be

$$P_G(z_1, \ldots, z_n) = |G|^{-1} \sum_{g \in G} {z_1}^{c_1(g)} \ldots {z_n}^{c_n(g)}, \tag{1}$$

where $c_i(g)$ denotes the number of cycles in g of length i. The key result just referred to is that the number of orbits of Σ^n under the action of G is P_G evaluated at the point $z_1 = \cdots = z_n = k$. For many important choices for G, this computation is feasible and leads to results concerning the number of unlabelled combinatorial structures of various kinds [11]. In this article we study the problem of computing P_G for an arbitrary permutation group G and at an arbitrary point.

1.2 Algorithmic preliminaries

It is clear that the cycle-index polynomial of a permutation group G may be computed at a specified point $z_1 = a_1, \ldots, z_n = a_n$, directly from definition (1) by explicit enumeration of the elements of G. In general, however, the order of G is exponentially larger than its description, say, as a list of generators. The interesting question from a computational viewpoint is whether some efficient algorithm exists for computing the cycle-index polynomial that avoids

[1] Note that the permutation g is considered to act on *positions* rather than *indices*, since this is perhaps the easier to grasp of the two possible conventions.

explicit enumeration. In the theoretical computer science tradition, we take as a first approximation to the notion of "efficient algorithm" a procedure (expressed as a Turing machine, or as a program in some simple programming language) that runs in time polynomial in some natural measure of input size.

In this article we take as our measure of input size simply the degree n of the permutation group G presented as input. This choice requires justification. On the one hand, a simple counting argument demonstrates that $\Omega(n \log n)$ bits are required to encode all permutation groups of degree n, so it is unreasonable to choose a measure of input size substantially smaller than n. On the other hand, as we shall see presently, every permutation group G of degree n has a simple and compact (in terms of n) encoding that allows many questions about G to answered efficiently.

Following Sims [19], for $1 \le i \le n$ let

$$G_i = \{g \in G : j^g = j, \text{ for all } 0 \le j < i\},$$

be the subgroup of G stabilising $[i]$ pointwise, and let $G_0 = G$. Then

$$G = G_0 \ge G_1 \ge \cdots \ge G_n = \{\mathbf{1}\}$$

is a sequence of subgroups of G with the property that the index $|G_i : G_{i+1}|$ of G_{i+1} in G_i is bounded by n. For $0 \le i \le n-1$, let U_i be a right transversal for $G_i : G_{i+1}$, i.e., a set that contains precisely one element from each right coset of G_{i+1} in G_i. The collection $\{U_i\}$ is called a *strong generating set* for G. Note that a strong generating set contains only $\mathrm{O}(n^2)$ permutations in total.

Aside from economy of space, a strong generating set has the advantage of efficiently supporting various operations, of which the most basic is membership testing. To decide whether permutation g is member of G, search for a permutation $u \in U_0$ such that gu^{-1} stabilises 0. If no such permutation exists, then $g \notin G$; otherwise the permutation u is unique, and $g \in G$ iff $gu^{-1} \in G_1$. Now recursively test whether $gu^{-1} \in G_1$ using the observation that $\{U_i : 1 \le i \le n-1\}$ is a strong generating set for G_1. Making realistic assumptions about the model of computation, the decision procedure just sketched can be implemented to run in time $\mathrm{O}(n^2)$.

Furst, Hopcroft, and Luks [7], who were the first to analyse Sims' data structure from a complexity-theoretic viewpoint, showed that a strong generating set may be computed in time $\mathrm{O}(n^6)$ from an arbitrary (small) set of generators for G. This bound on time-complexity was subsequently improved to $\mathrm{O}(n^5)$ by the author [12] using elementary techniques, and to $\widetilde{\mathrm{O}}(n^4)$ by Babai, Luks, and Seress [2], using non-elementary techniques relying on the classification of finite simple groups.[2]

---

[2]The $\widetilde{\mathrm{O}}()$ notation hides not merely constants, but also arbitrary powers of $\log n$.

1.3 Complexity-theoretic preliminaries

Let Δ be a finite alphabet, perhaps the binary alphabet, in which we agree to encode problem instances, whether they be groups, graphs, numbers, or whatever. We are concerned with computational problems whose solution is a natural number; abstractly, such problems may be viewed as functions from instances Δ^* to solutions $\mathbb{N}$. In fact, for reasons that will become apparent shortly, it will prove convenient to consider a slightly more general set-up in which the instance consists of a number m of recognisable parts, all encoded over the alphabet Δ.

We say that a function $f : (\Delta^*)^m \to \mathbb{N}$ is *polynomial time* if there exists a Turing machine that computes $f(\alpha_1, \ldots, \alpha_m)$ in time polynomial in $|\alpha_1| + \cdots + |\alpha_m|$. The complexity class #P was introduced by Valiant [22, 23] as a counting analogue to the more familiar class NP of decision problems. It may be defined in several equivalent ways, of which the following is particularly well suited to our purpose. A function $f : \Delta^* \to \mathbb{N}$ is contained in #P precisely if it can be expressed in the form

$$f(x) = \sum_{y \in \Delta^{p(|x|)}} w(x, y), \tag{2}$$

where p is a univariate polynomial, and $w : (\Delta^*)^2 \to \mathbb{N}$ is a polynomial-time "weight function."

For example, we might interpret the word x as the encoding of an undirected graph G, the word y as the encoding of a subgraph of G, and define $w(x, y)$ to be 1 if y encodes a perfect matching in G, and 0 otherwise. Then the function f defined by (2) counts the number of perfect matchings in a graph. Since the function w is clearly polynomial time, we may deduce that the problem of counting perfect matchings in a graph is in the class #P. As the specific combinatorial structure "perfect matching" in the above example could be replaced by almost any other (positively weighted) structure, it will be apparent that #P is a wide ranging class, which contains many problems of combinatorial interest.

Just as with NP, the class #P contains "complete" functions that are efficiently universal for the class, and hence (informally) as hard to compute as any function in the class. A function f' is *polynomial-time Turing reducible to* f if there is a polynomial-time Turing machine that computes f' given an oracle for f.[3] A function f is #P-*hard* if every function in #P is Turing reducible to f; it is #P-*complete* if, in addition, $f \in$ #P. The key observation is that if *any* #P-hard function is computable in polynomial time, then *every* function in #P is computable in polynomial time. Since #P is a wide-ranging

[3] An *oracle* for a function f is a black box that takes as input a word $x \in \Delta^*$ and in one time-step produces as output $f(x)$.

class containing many apparently hard-to-compute functions, #P-hardness (or completeness) may be interpreted as strong evidence for intractability.

Valiant [22] showed that counting perfect matchings in a bipartite graph is #P-complete and hence likely to be computationally intractable, even though there is a polynomial-time procedure for deciding existence of a perfect matching, using the classical "augmenting path" technique. This was the first really convincing demonstration of a phenomenon that is now recognised as widespread: that a combinatorial counting problem may be computationally intractable, even when the allied decision problem is computationally easy. The catalogue of known #P-complete problems is now very extensive.

2 The complexity of counting modulo a group of symmetries

For an infinite sequence $(a_i : i \in \mathbb{N}^+)$ of rational numbers, let $\textsc{CycleIndex}(a_i)$ be the problem:

Instance (an encoding of) a permutation group G of degree n;

Output the value $|G|\, P_G(a_1, \ldots, a_n)$.

The factor $|G|$ in the output makes the problem into a straight generating function evaluation. Its addition is a matter of convenience, and has no essential effect on the computational difficulty of the problem, as the order $|G|$ of a group G is easy to compute from a strong generating set for G. It is straightforward to verify, directly from the definition of the class #P, that $\textsc{CycleIndex}(k, k, \ldots) \in \#\mathrm{P}$, for any positive integer k.

Three cases are known in which $\textsc{CycleIndex}(a_i)$ is easy to compute:

(i) if $a_i = 0$ for all $i > 1$, then only the identity permutation carries any weight and the required output is a_1^n;

(ii) if (a_i) is a geometric progression with initial element r and common ratio r, then every permutation carries equal weight, and the required output is $|G|\, r^n$;

(iii) if (a_i) is a geometric progression with initial element r and common ratio $-r$, then all even (respectively, odd) permutations carry weight r^n (respectively, $-r^n$), and the required output is either 0 or $|G|\, r^n$ depending on whether or not G contains odd permutations.

Note that all three cases are algorithmically trivial. Goldberg [9, Thm 22] has shown that for "almost all" non-negative sequences (a_i) not covered by the above cases (for details see Theorem 2), CycleIndex(a_i) is #P-hard, and hence likely to be computationally intractable. The proof of Theorem 2 is technically intricate, so we content ourselves here with proving a special case of particular interest, namely ($a_i = 2$). In doing so, we hope to illustrate the concepts introduced in Section 1.3, and gain insight into why the cycle-index polynomial is hard to compute.

Theorem 1 CycleIndex$(2, 2, \ldots)$ *is* #P-*complete.*

Proof The following problem, which we refer to as #IndSet, was included in Valiant's original list [23] of #P-complete problems, in the guise of "Monotone 2-Sat:"

Instance an undirected graph Γ;

Output the number of independent sets (not necessarily maximal) in Γ.

Our proof strategy is to exhibit a polynomial-time Turing reduction from #IndSet to CycleIndex$(2, 2, \ldots)$. Since the relation "is polynomial-time Turing reducible to" is transitive, it will follow that CycleIndex$(2, 2, \ldots)$ is #P-complete.

Let $\Gamma = (V, E)$ be an undirected graph with vertex set V and edge set E, viewed as an instance of #IndSet, and let $n = |V|$ and $m = |E|$. Arbitrarily orient the edges of G so that each edge $e \in E$ has a defined start vertex e^- and end vertex e^+. Construct a permutation group $G = G(\Gamma)$ of degree $4m$ on base set

$$\Omega = \Big\{\langle e, 00\rangle, \langle e, 01\rangle, \langle e, 10\rangle, \langle e, 11\rangle : e \in E\Big\}$$

as follows. For $e \in E$, define (in cycle notation)

$$h_e = (\langle e, 00\rangle, \langle e, 01\rangle)\,(\langle e, 10\rangle, \langle e, 11\rangle)$$

and

$$h'_e = (\langle e, 00\rangle, \langle e, 10\rangle)\,(\langle e, 01\rangle, \langle e, 11\rangle).$$

Then $G = \langle g_v : v \in V\rangle$, where, for each $v \in V$, the generator g_v is given by

$$g_v = \prod_{e:e^-=v} h_e \prod_{e:e^+=v} h'_e.$$

Observe that the generators g_v commute and have order 2. It follows that every group element $g \in G$ may be expressed in the form $g = g_U = \prod_{u\in U} g_u$, for some $U \subseteq V$. It is easy to see that the set U is uniquely determined by g, and hence that $|G| = 2^n$.

For any $U \subseteq V$, the cycle structure of g_U is $1^{4i} \cdot 2^{2(m-i)}$ where i is the number of edges in Γ with both endpoints in the complement of U. Denote by N_i the number of subsets $U \subseteq V$ such that precisely i edges of Γ have both endpoints in the complement of U. Note that N_0 is the number of vertex covers in Γ (and hence, by complementation, of independent sets). Then, for j a positive integer,

$$|G|\, P_G(2^j, 2^j, \ldots, 2^j) = \sum_{i=0}^{m} N_i\, 2^{2(m+i)j} = 2^{2mj} p(2^{2j}),$$

where p is the polynomial $p(\xi) = \sum_{i=0}^{m} N_i \xi^i$. If the values $|G|\, P_G(2^j, \ldots, 2^j)$ for $j = 1, \ldots, m+1$ were known, it would be possible to interpolate to determine $p(0) = N_0$. But

$$P_G(2^j, 2^j, \ldots, 2^j) = P_{G \wr \{\mathbf{1}_j\}}(2, 2, \ldots, 2),$$

where $G \wr \{\mathbf{1}_j\}$ denotes the wreath (Kranz) product of G with the trivial permutation group on j letters.

Thus, by making $m+1$ calls to an oracle for CycleIndex$(2, 2, \ldots)$ and performing an interpolation of a univariate polynomial of degree m, it is possible to evaluate the number of independent sets in the graph Γ. The procedure just described constitutes a polynomial-time Turing reduction from #IndSet to CycleIndex$(2, 2, \ldots)$. $\square$

As indicated earlier, Theorem 1 is a special case of a general result of Goldberg.

Theorem 2 *Let (a_i) be a sequence of non-negative rational numbers satisfying $a_i \neq a_1^i, 0$, for some i. Then* CycleIndex(a_i) *is #P-hard.*

A proof may be found in [9, p. 150] or [10]. Note that we cannot claim #P-completeness here; however, if we assume in addition that (a_i) is a sequence of natural numbers computable in time polynomial in i, then CycleIndex(a_i) is in #P, and hence #P-complete.

It is reasonable to conjecture that the problem CycleIndex(a_i) is #P-hard for all sequences not covered by cases (i)–(iii) identified at the beginning of the section, but this is not known for certain. There is unlikely to be a major conceptual difficulty in resolving the issue, but the solution may require a case analysis of some technical complexity. Note that Theorem 2 already covers all non-trivial non-negative sequences, except those obtained from a geometric progression by setting an arbitrary set of elements to 0.

3 Approximate counting and uniform sampling

We have seen that in almost all its non-trivial variants, the problem of evaluating the cycle-index polynomial P_G *exactly* is #P-hard. This should perhaps not surprise us, since the problem of determining whether a permutation group contains a permutation with a specified cycle structure is known to be NP-complete. However, we might reasonably ask whether there is an efficient *approximation* algorithm for P_G. This is one of a group of three related questions that may be formalised as follows. (As before Σ is a finite alphabet of cardinality k.)

(a) Is there a *fully polynomial randomised approximation scheme* [17] (fpras) for estimating $P_G(k, \dots, k)$? That is to say, is there a randomised algorithm that takes as input a group G and $\varepsilon > 0$, and produces as output a number Y (a random variable) such that

$$\Pr\Big((1-\varepsilon)P_G(k,\dots,k) \le Y \le (1+\varepsilon)P_G(k,\dots,k)\Big) \ge \frac{3}{4},$$

and, moreover, does so within time $\mathrm{poly}(n, \varepsilon^{-1})$?

(b) Is there a polynomial-time *almost uniform sampler*[4] [16] for the orbits of Σ^n under the action of G? That is to say, is there a randomised algorithm that takes as input a group G and $\varepsilon > 0$, and produces as output a word $Y \in \Sigma^n$ (a random variable) such that for each orbit O,

$$(1-\varepsilon)N^{-1} \le \Pr(Y \in O) \le (1+\varepsilon)N^{-1},$$

where $N = P_G(k, \dots, k)$ is the total number of orbits? The execution time is required to be bounded by $\mathrm{poly}(n, \varepsilon^{-1})$.

(c) Is there a polynomial-time "almost-w" sampler for G, where the weight function $w : G \to \mathbb{N}$ is defined by $w(g) = k^{c(g)}$, where $c(g)$ denotes the number of cycles in g. That is to say, is there a randomised algorithm that takes as input a group G and $\varepsilon > 0$, and produces as output a permutation $Y \in G$ (a random variable) such that for each $g \in G$,

$$(1-\varepsilon)w(g)Z^{-1} \le \Pr(Y = g) \le (1+\varepsilon)w(g)Z^{-1},$$

where $Z = |G|\, P_G(k, \dots, k)$? Again, the execution time is required to be bounded by $\mathrm{poly}(n, \varepsilon^{-1})$.

The complexity of approximate counting and of almost uniform sampling are known to be closely related,[5] which would lead one to suppose that questions

[4] When this concept was first introduced, "generator" was used in place of "sampler", but the latter word is more specific.

[5] A rather precise statement of this relationship has been formulated by Jerrum, Valiant, and Vazirani [16].

(a), (b), and (c) ought to be equivalent. However, the situation here is atypical, and it is not clear, for example, whether resolving question (a) in the affirmative would immediately settle either of the others. The two known entailments are described in the following proposition, whose (routine) proof may be found in [13, 14].

Proposition 3 *An affirmative answer to question* (c) *would entail affirmative answers to* (a) *and* (b)*.*

I feel that questions (a)–(c) are quite significant, and they are all unresolved. It is worth considering briefly why the proof technique (i.e., reduction) of Theorem 1 cannot be used to provide a negative answer to question (a); after all, there is convincing evidence (see [21, Thm 1.17]) that there is no fpras for #INDSET, which is the starting point for the reduction. The catch is that the reduction makes essential use of polynomial interpolation, and interpolation does not preserve closeness of approximation.

Question (a) concerns the evaluation of P_G at points of the form $z_1 = z_2 = \cdots = z_n = k$, for some positive integer k. While these are undoubtedly the points of greatest combinatorial interest, it is natural to enquire whether it is possible to approximate P_G at other points. Interestingly, relaxing either constraint — that the z_i are assigned equal values, or that they are assigned integer values — appears to lead to computational intractability. We explore this phenomenon in the following subsection.

3.1 Negative results

The first result provides evidence that evaluating the cycle-index polynomial at points whose coordinates are not all equal is computationally intractable. Before presenting the result, we need to consider the nature of the evidence. RP is the class of decision problems that can be solved in polynomial time by a certain type of randomised algorithm which is allowed one-sided errors. (See [3, p. 138] for a precise definition.) Rabin's celebrated primality test is an example of such an algorithm. The class RP is widely regarded as a reasonable extension to the class P, which preserves the notion of efficient solvability. It would be surprising if all NP-complete problems were efficiently solvable in the RP sense, and so it is conjectured that RP is strictly contained in NP. Goldberg [9, 10] has shown the following.

Theorem 4 *Let* (a_i) *be a sequence of non-negative rationals, such that there exists* i *with* $a_i > a_1^i$. *There can be no fpras for* CYCLEINDEX(a_i)*, unless* RP = NP.

The above theorem is couched in slightly different language to [9, Thm 23] and [10, Thm 3], but the essential content is the same. The proof given in those references is adequate to establish this version of the theorem. As an immediate corollary of Theorem 4 we have:

Corollary 5 *Assuming* RP $\neq$ NP, *there is no fpras for* CycleIndex$(0, 2, 0, 2, \ldots)$.

In the light of the earlier discussion of the relationship between RP and NP, it seems unlikely that the cycle-index polynomial can be efficiently approximated at the point $0, 2, 0, 2, \ldots$. The combinatorial significance of Corollary 5 is that $P_G(0, 2, 0, 2, \ldots)$ counts the number of self-complementary orbits of the action of G on $\{0, 1\}^n$; an orbit is *self-complementary* if it is invariant under the interchange of letters 0 and 1. In this manner, the number of self-complementary graphs (graphs that are isomorphic to their own complements) may be expressed as an evaluation of the cycle-index polynomial.

Given the lack of progress on question (a), the following result of Goldberg and Jerrum may come as a surprise.

Theorem 6 *Let a be a positive rational that is not an integer. There can be no fpras for* CycleIndex$(a, a, \ldots)$, *unless* RP = NP.

It is necessary to explain the phenomenon that allows the all important integer case to escape from the grasp of the proof of Theorem 6. Suppose a is as in the statement of Theorem 6, and let $h = \lceil a \rceil + 1$. Then it can be shown that

$$P_{\mathrm{Alt}[h]}(a, \ldots, a) < P_{\mathrm{Sym}[h]}(a, \ldots, a), \tag{3}$$

where $\mathrm{Sym}[h]$ and $\mathrm{Alt}[h]$ denote, respectively, the symmetric and alternating groups on $[h]$. In contrast, if a is a positive integer, inequality (3) must fail, as can be seen by considering the combinatorial interpretations of $P_{\mathrm{Alt}[h]}(a, \ldots, a)$ and $P_{\mathrm{Sym}[h]}(a, \ldots, a)$ in the integral case.

To exploit inequality (3) in the case that a is not an integer, we adapt the reduction of Theorem 1 as follows. Given a graph $\Gamma = (V, E)$ with n vertices and m edges, we again construct a permutation group G, this time of degree hm. The base set of G is $\Omega = E \times [h]$, and we start the construction of the generating set of G by adding for each $e \in E$, a set of permutations that generate the alternating group $\mathrm{Alt}(\{e\} \times [h])$ on $\{e\} \times [h]$. As in the proof of Theorem 1 we introduce, for each vertex $v \in V$, a permutation g_v that acts on all sets $\{e\} \times [h]$ for which the edge e is incident at vertex v. The action of g_v on $\{e\} \times [h]$ is simply to transpose an arbitrary pair of elements.

As before there is a natural correspondence between subsets $U \subseteq V$ and objects in the group G; in this case, each vertex subset U corresponds not to a single element in G, but to a coset of $\prod_{e \in E} \mathrm{Alt}(\{e\} \times [h])$ in G; conversely, each coset corresponds to two vertex subsets, U and $\overline{U}$, that are complements of each other. In other words, there is a bijection between cosets and bipartitions $\{U, \overline{U}\}$ of the vertex set of Γ. Each bipartition $\{U, \overline{U}\}$ defines a *cut* containing all edges of Γ with one endpoint in U and one in $\overline{U}$. When the cycle-index polynomial of G is partitioned according to the cosets, by inequality (3), it is those cosets corresponding to maximum (cardinality) cuts that carry the greatest weight.

Now by employing the wreath product construction from the proof of Theorem 1, we may efficiently boost the weight of the cosets corresponding to maximum weight cuts until they account for almost all of the weight of the cycle-index polynomial. If this is done, then an approximation to the value of the cycle-index polynomial will yield the size of a maximum cut, and also an approximation to the number of such cuts. However, determining whether a graph Γ has a cut of cardinality at least C is an NP-complete problem [8, p. 210]. The upshot is that a fpras for $\textsc{CycleIndex}(a, a, \ldots)$ would entail the existence of a fast randomised (RP-style) algorithm for a particular NP-complete problem, and hence for all problems in NP.

When a is an integer, one can have, at best, equality in (3). The reverse inequality is possible and corresponds to the "minimum cut" problem which, in the absence of further constraints, is trivial. It is hoped that the above account, though telegraphic, does give an intuitive feel for the phenomenon of intractability of the cycle-index polynomial at non-integer points.

3.2 Positive results

As we have observed, there is no known general and efficient procedure for estimating the cycle-index polynomial at integer points, or for sampling uniformly at random (u.a.r.) from the orbits of the action of G on Σ^n. However there is a promising approach to both of these problems, which will be described in this final subsection. We concentrate on the latter problem; any success that the method may find there easily translates to the former problem

Our approach to sampling orbits is to simulate an appropriately defined Markov chain. This technique has proved fruitful on a number of occasions in recent years; previous applications include an algorithm of Jerrum and Sinclair for estimating the permanent of a 0,1-matrix [15] and one of Dyer, Frieze, and Kannan for estimating the volume of a convex body in n-dimensional space [6]. In this instance we wish to construct a Markov chain $M = M(G, \Sigma)$ whose state space is Σ^n, and whose stationary distribution assigns equal probability to each orbit. In fact, we shall aim at something stronger, namely, a stationary distribution that assigns to each word $\alpha \in \Sigma^n$ a probability inversely proportional to the size of the orbit α^G containing α.

The transition probabilities from a state $\alpha \in \Sigma^n$ are specified by the following conceptually simple two-step experiment:

(1) choose g uniformly at random (u.a.r.) from the point stabiliser $G_\alpha = \{g \in G : \alpha^g = \alpha\}$;

(2) choose β u.a.r. from the set $\{\beta \in \Sigma^n : \beta^g = \beta\}$.

The new state is β.[6] Before analysing the stationary distribution of M, we should pause to consider the computational complexity of implementing the above experiment. Step (2) is computationally undemanding, and amounts to assigning, u.a.r. and independently, a symbol from Σ to each cycle of g. Step (1) is more interesting, and is equivalent (under randomised polynomial-time reducibility) to computing a setwise stabiliser in a permutation group, a task that includes deciding isomorphism of two graphs as a special case. The computational complexity of the setwise stabiliser problem is open: it is one of the very rare natural candidates for a problem that is in the class NP, but is neither in P nor NP-complete.

Although no general polynomial-time algorithm for implementing step (1) is known, and it is perfectly possible that none exists, there are significant classes of groups G for which step (1) does have an efficient implementation. Luks has shown that p-groups — groups in which every element has order a power of p for some prime p — is an example of such a class [18].

Returning to the Markov chain itself, we note immediately that M is ergodic, since every state can be reached from every other in a single transition, by selecting the identity permutation in step (1). The easiest way to get at the stationary distribution is perhaps by considering a random walk on the bipartite graph B that has vertex bipartition (Σ^n, G) and edge set $\{(\alpha, g) : \alpha^g = \alpha\}$. It is clear that the Markov chain M may be viewed as sampling the random walk on B after every even step. Let $\pi : \Sigma^n \to [0,1]$ denote the stationary distribution of M. Then $\pi(x)$ is proportional to the degree of vertex α in the graph B, which is $|G_\alpha|$. It is an elementary group-theoretic fact that $|G_\alpha| \times |\alpha^G| = |G|$, and hence $\pi(\alpha)$ is inversely proportional to $|\alpha^G|$. We have therefore established:

Theorem 7 *Let π be the stationary distribution of Markov chain M. Then $\pi(\alpha) = |\alpha^G|^{-1} P_G(k, \ldots, k)^{-1}$ for all $\alpha \in \Sigma^n$; in particular, π assigns equal probability to each orbit α^G.*

Dually we might consider the Markov chain M' with state space G and transition probabilities modelled by an experiment in which steps (1) and (2) appear transposed. By relating M' to the random walk on B we easily obtain:

[6] Peter Cameron has observed that this process is defined for any group action, not just the special case of G acting on Σ^n by the permutation of positions.

Theorem 8 *Let π' be the stationary distribution of Markov chain M'. Then $\pi'(g) = k^{c(g)}|G|^{-1}P_G(k,\ldots,k)^{-1}$ for all $g \in G$, where $c(g)$ denotes the number of cycles in the permutation g.*

Note that the stationary distributions of the Markov chains M and M' match the sampling distributions specified in questions (b) and (c).

Although the topic of random walks on groups has received much attention (see for example the work of Aldous [1] and Diaconis [5]), previous authors have been concerned with walks which converge to a uniform distribution. The novel aspect of the current investigation is that the stationary distribution is required to be highly non-uniform.

We have seen that the stationary distribution of the Markov chain M is appropriate for sampling u.a.r. from the set of orbits of the G-action on Σ^n. For the Markov chain simulation approach to be computationally efficient, it is necessary for M to be *rapidly mixing*. We give a precise meaning to this informal requirement by insisting that M should be "close" to stationarity after a number of steps that is bounded by a polynomial in n. Since the size of the state space is exponential in n, this is a non-trivial requirement.

There are a number of ways of quantifying "closeness" to stationarity, but they are all essentially equivalent in this application. Consider an ergodic Markov chain with state space Q. Let $q \in Q$ be an arbitrary state, and denote by $P^t(q,\cdot\,)$ the distribution of the state at time t given that q is the initial state. Let π be the stationary distribution of M. Then the *variation distance* at time t with respect to the initial state q is defined to be

$$\delta_q(t) = \max_{S\subseteq Q}\left|P^t(q,S) - \pi(S)\right| = \tfrac{1}{2}\sum_{q'\in Q}\left|P^t(q,q') - \pi(q')\right|.$$

The rate of convergence may be measured by the function

$$\tau_q(\varepsilon) = \min\{t : \delta_q(t') \le \varepsilon \text{ for all } t' \ge t\}.$$

The Markov chain simulation technique will yield an efficient almost uniform sampler for orbits of the G-action on Σ^n provided the Markov chain $M(G,\Sigma)$ mixes rapidly. In recent years, techniques have been developed for bounding the mixing rate of combinatorially defined Markov chains, using, among other ideas, the relation between mixing rate and the expansion properties of the Markov chain viewed as a graph. Sinclair has provided a useful survey of these techniques, in addition to presenting some sharpened bounds [20]. Nevertheless, proofs of rapid mixing still tend to be technically involved.

Since no counterexamples have been identified, it remains a possibility that for any fixed alphabet Σ, the Markov chain $M(G,\Sigma)$ is rapidly mixing for all choices of G; specifically, that there is a *fixed* polynomial (in n and $\log\varepsilon^{-1}$)

that uniformly bounds $\tau_q(\varepsilon)$ for all possible groups G. If this were so — and the author would not hazard a conjecture on this point — then questions (b) and (c) would be answered in the affirmative, at least for groups for which the Markov chain could be efficiently simulated. Note that by Proposition 3, we would also have, indirectly, an affirmative answer to question (a).

A proof that the Markov chain $M(G,\Sigma)$ is rapidly mixing, even for the special case of G an abelian p-group, would have a significant algorithmic consequence in the field of statistical physics: it would imply the existence of a polynomial-time algorithm for estimating the partition function of a ferromagnetic p-state Potts system. We should therefore not be surprised if progress on the rapid mixing question for general permutation groups is rather slow. A more realistic programme for the short term is to produce a catalogue of families of groups G for which rapid mixing of $M(G,\Sigma)$ can be rigorously demonstrated. The following two results represent a modest start on this programme. Their proofs, which employ standard techniques, can be found in [14]. (The author is grateful to David Aldous for suggesting an improvement to the proof of Proposition 9, leading to a tighter bound on convergence time.)

Proposition 9 *Let* $\mathrm{Sym}[n]$ *denote the symmetric group on* $[n]$, *and* $M = M(\mathrm{Sym}[n],\Sigma)$ *the derived Markov chain defined earlier in the subsection. The convergence time of* M *is*

$$\tau_\alpha(\varepsilon) = \mathrm{O}(\log n\varepsilon^{-1}),$$

uniformly over the choice of initial state $\alpha \in \Sigma^n$.

Note that convergence is *very* rapid in the case of the full symmetric group. The constant implicit in the O-notation is dependent of the size of the alphabet Σ; for $\Sigma = \{0,1\}$, the constant is 1, provided the logarithm is taken to base 2.

Proposition 10 *Let* G *denote any cyclic permutation group on* $[n]$, *and* $M' = M'(G,\Sigma)$ *the derived Markov chain defined earlier in the subsection. The convergence time of* M' *is*

$$\tau_g(\varepsilon) = \mathrm{O}(n \log n \log \varepsilon^{-1}),$$

uniformly over the choice of initial state $g \in G$.

The constant implicit in the O-notation is here independent of Σ.

Acknowledgement

I thank Leslie Goldberg for helpful comments on a draft of this article.

References

[1] David Aldous, Random walks on finite groups and rapidly mixing Markov chains, *Séminaire de Probabilités XVII 1981/82* (A. Dold and B. Eckmann, eds), Springer Lecture Notes in Mathematics **986**, pp. 243–297.

[2] László Babai, Eugene Luks, and Ákos Seress, Fast management of permutation groups, *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, Computer Society Press, 1988, pp 272–282.

[3] J. L. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity, Volume I*, Springer-Verlag, Berlin, 1988.

[4] N. G. de Bruijn, Pólya's theory of counting, *Applied Combinatorial Mathematics*, E. F. Beckenbach (ed.), John Wiley and Sons, 1964, pp. 144–184.

[5] Persi Diaconis, *Group representations in probability and statistics*, Institute of Mathematical Statistics, Hayward CA, 1988.

[6] Martin Dyer, Alan Frieze, and Ravi Kannan, A random polynomial time algorithm for approximating the volume of convex bodies, *Journal of the ACM* **38** (1991) pp. 1–17.

[7] Merrick Furst, John Hopcroft, and Eugene Luks, Polynomial time algorithms for permutation groups, *Proceedings of the 21st IEEE Symposium on Foundations of Computer Science*, Computer Society Press, 1980, pp. 36–41.

[8] Michael R. Garey and David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[9] Leslie Ann Goldberg, *Efficient Algorithms for Listing Combinatorial Structures*, Cambridge University Press, 1993.

[10] Leslie Ann Goldberg, Automating Pólya theory: the computational complexity of the cycle index polynomial, *Information and Computation* **105** (1993), pp. 268–288.

[11] Frank Harary and Edgar M. Palmer, *Graphical enumeration*, Academic Press, 1973.

[12] Mark Jerrum, A compact representation for permutation groups, *Journal of Algorithms* **7** (1986) pp. 60–78.

[13] Mark Jerrum, Uniform sampling modulo a group of symmetries using Markov chain simulation, *Expanding Graphs*, Joel Friedman (ed.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science **10**, American Mathematical Society, 1993, pp. 37–47.

[14] Mark Jerrum, *Uniform sampling modulo a group of symmetries using Markov chain simulation.* Report ECS-LFCS-93-272, Department of Computer Science, University of Edinburgh, July 1993.

[15] Mark Jerrum and Alistair Sinclair, Approximating the permanent, *SIAM Journal on Computing* **18** (1989) pp. 1149–1178.

[16] M. R. Jerrum, L. G. Valiant and V. V. Vazirani, Random generation of combinatorial structures from a uniform distribution, *Theoretical Computer Science* **43** (1986), pp. 169–188.

[17] Richard M. Karp and Michael Luby, Monte-Carlo algorithms for enumeration and reliability problems, *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, Computer Society Press, 1983, pp. 56–64.

[18] Eugene M. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time, *Journal of Computer and System Sciences* **25** (1982), pp. 42–65.

[19] Charles C. Sims, Computational methods in the study of permutation groups, *Computational Problems in Abstract Algebra*, J. Leech (ed.), Pergamon Press, New York, 1970, pp. 169–183.

[20] Alistair Sinclair, Improved bounds for mixing rates of Markov chains and multicommodity flow, *Combinatorics, Probability and Computing* **1** (1992), pp. 351–370.

[21] A. J. Sinclair, *Randomised algorithms, for counting and generating combinatorial structures*, Advances in theoretical computer science, Birkhäuser, Boston, 1993.

[22] L. G. Valiant, The complexity of computing the permanent, *Theoretical Computer Science* **8** (1979), pp. 189–201.

[23] Leslie G. Valiant, The complexity of enumeration and reliability problems, *SIAM Journal on Computing* **8** (1979), pp. 410–421.