The computational complexity of counting*

Mark Jerrum†
Department of Computer Science
University of Edinburgh

28th July 1994

Abstract

The complexity theory of counting contrasts intriguingly with that of existence or optimisation.

1 Counting versus existence

The branch of theoretical computer science known as *computational complexity* is concerned with quantifying the computational resources required to achieve specified computational goals. Classically, the goal is often to decide the existence of a certain combinatorial structure, for example, whether a given graph G contains a Hamilton cycle. Alternatively, the goal might be to find an occurrence of the structure that is optimal with respect to a certain measure; in the context of the structure "Hamilton cycle," the notorious *Travelling Salesman Problem* may be cited as an example. Less well studied, and somewhat less well understood, are counting problems, such as determining how many Hamilton cycles a graph G contains. In some areas, such as statistical physics, counting problems arise directly; in many others they appear in the guise of discrete approximations to continuous problems involving multivariate integration. This article aims to sketch the complexity theory of counting, highlighting the ways in which it diverges from that of existence or optimisation.

Let Σ be an alphabet, possibly the binary alphabet, in which the objects of interest (e.g., graphs and Hamilton cycles) may be encoded. A *witness-testing predicate* for some combinatorial structure $\mathcal{S}$ is a predicate $\psi : \Sigma^* \times \Sigma^* \to \{0, 1\}$,

*To be presented at the International Congress of Mathematicians, Zürich, August 1994.

†The author is a Nuffield Foundation Science Research Fellow, and 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; e-mail: `mrj@dcs.ed.ac.uk`.

where the truth of $\psi(x, y)$ is to be interpreted as "y is an occurrence of structure $\mathcal{S}$ within instance x." Specialising to the structure "Hamilton cycle," $\psi(x, y)$ would be true precisely if the words x and y encode (respectively) a graph G and a subgraph H of G, and H is a Hamilton cycle in G. The existence predicate $\phi(x)$ for the structure $\mathcal{S}$ may be expressed as

$$\phi(x) \Leftrightarrow \exists\, y \in \Sigma^* \left[|y| = p(|x|) \wedge \psi(x, y)\right], \tag{1}$$

where p is a polynomial that depends on the exact encodings used. In our running example, ϕ is the "Hamiltonicity predicate," so that $\phi(x)$ expresses the situation that x encodes a Hamiltonian graph. For convenience, we abbreviate the relationship (1) between ϕ and ψ to $\phi(x) \Leftrightarrow \exists^p y\, \psi(x, y)$.

In general, the witness-testing predicate $\psi(x, y)$ belongs to the class P of polynomial-time predicates, i.e., those that are computable in time polynomial in the lengths of their arguments. (Observe, for example, that it is easy to determine whether a given subgraph H of a graph G is a Hamilton cycle of G, and that the same is true of most other combinatorial structures we might consider.) The class NP (which might more suggestively be denoted $\exists$P) contains precisely those predicates ϕ that may be derived from polynomial-time two-place predicates ψ via (1). It is clear that if $\phi \in$ NP then $\phi(x)$ may be decided in time exponential in $|x|$ by exhaustive search over all words $y \in \Sigma^*$ of length polynomial in $|x|$. The key issue from a complexity-theoretic viewpoint is whether $\phi(x)$ may be decided in much less than exponential time, in particular, whether $\phi \in$ P.

It is important to note that the *complexity classes* P and NP (and indeed all the others we shall encounter in this article) are invariant under "reasonable" variation in the computational model and in the choice of encodings. Thus, it is of no great significance whether computation is modelled by a Turing machine or something more akin to existing computers, nor whether graphs are encoded as adjacency matrices or as incidence lists. A polynomial-time computation will remain polynomial-time, even though the degree of the polynomial may change.

It is clear that P $\subseteq$ NP, but strictness of the inclusion remains a major open question. Under the assumption P $\subset$ NP — which is widely conjectured to be the case — it is possible to exhibit "natural" predicates in NP $-$ P using the notion of polynomial-time reducibility. A predicate ϕ' is said to be *polynomial-time (many-one or Cook-Karp) reducible to* ϕ if there exists a polynomial-time-computable function $g : \Sigma^* \to \Sigma^*$ satisfying $\phi'(x) \Leftrightarrow \phi(g(x))$. Many natural predicates $\phi \in$ NP, including Hamiltonicity, have the property that every predicate in NP is polynomial-time reducible to ϕ, and such predicates are said to be *NP-complete*. It is an easy consequence of the definitions that no NP-complete predicate can be in P, unless P = NP. Thus, NP-completeness of a predicate such as Hamiltonicity can be regarded as evidence of computational intractability.

The framework just described extends smoothly to counting problems. The counting function $f : \Sigma^* \to \mathbb{N}$ associated with witness-testing predicate ψ is

$$f(x) = \left|\left\{y \in \Sigma^* : |y| = p(|x|) \wedge \psi(x, y)\right\}\right|; \tag{2}$$

for convenience, we abbreviate this relationship to $f(x) = \#^p y\, \psi(x, y)$. The class #P, the counting analogue of NP, contains precisely those functions f that can be derived from polynomial-time two-place predicates ψ via (2). As with NP, the class #P contains complete problems, which are computationally the hardest problems in #P relative to polynomial-time reducibility. However, to obtain a rich class of #P-complete problems it is necessary to employ a somewhat more liberal notion of reducibility known as Turing reducibility. A function f' is *Turing reducible to* f if there is a polynomial-time algorithm that computes f' given an an oracle for f.[1] A function $f \in \#\mathrm{P}$ is #P-*complete* if every function in #P is Turing reducible to f.

It is clear that a procedure for counting witnesses must in particular decide between no witnesses and some, and hence counting is computationally at least as hard as deciding existence. The first evidence that counting can be harder than deciding existence for significant natural problems was provided by Valiant [18], who exhibited a natural witness-checking predicate for which the associated counting function is #P-complete, even though the associated existence predicate is in P. A *perfect matching* in a graph G is a subset M of the edge set of G such that every vertex of G is the endpoint of precisely one edge in M. Valiant showed that counting the number of perfect matchings in a bipartite graph (equivalently, evaluating the permanent of a 0,1-matrix) is #P-complete and hence likely to be computationally intractable, whereas deciding whether a bipartite graph contains a perfect matching (equivalently, deciding whether the permanent of a 0,1-matrix is non-zero) is in P, by virtue of the classical "augmenting path" algorithm.

From Valiant's initial collection [19], the catalogue of #P-complete counting problems has grown steadily. A significant recent contribution is Brightwell and Winkler's proof [2] that counting linear extensions of a partial order is #P-complete, settling a decade-old question. Perhaps the most extensive investigation in this direction was initiated by Jaeger, Vertigan and Welsh [6], who considered the computational complexity of evaluating the two-variable Tutte polynomial $T(M; x, y)$ of a matroid M, at all points and along significant curves in the x, y-plane. An astonishing variety of counting problems connected with matroids in general, and graphs in particular, may by viewed as evaluations of the Tutte polynomial at certain points and along certain curves. Many complexity-theoretic results may be read off from the classification developed in [6] and succeeding articles.

Empirically it appears to be the case that counting occurrences of a combinatorial structure is #P-complete whenever deciding existence of the structure is NP-complete. This apparent entailment may be formalised, leading to the following precise open question. Let ψ be any polynomial-time witness-testing predicate; is it the case that the function $f(x) = \#^p y\, \psi(x, y)$ is #P-complete whenever the predicate $\phi(x) = \exists^p y\, \psi(x, y)$ is NP-complete? There seems to be no good reason

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

why the answer should be yes in general, even though it is for all "natural" witness-checking predicates ψ that have been considered. Note that Valiant's result [18] assures us that the converse cannot hold unless P = NP.

2 Structural considerations

From the point of view of "structural complexity" it is natural to consider where the counting class #P lies in relation to other complexity classes. Certain relationships, for example that #P is contained in the class FPSPACE of functions computable in polynomial space, follow immediately from the definitions. A far from obvious and more enlightening containment was discovered about five years ago. Let us first extend the notation of Section 1 in natural way. For $\mathcal{C}$ a class of predicates, let $\exists\mathcal{C}$ denote the derived class obtained by the following construction. A k-place predicate ϕ is contained in $\exists\mathcal{C}$ if there is a $(k+1)$-place predicate $\psi \in \mathcal{C}$ and a polynomial p such that

$$\phi(x_1, \ldots, x_k) \Leftrightarrow \exists y \in \Sigma^* \Big[|y| = p(|x_1|, \ldots, |x_k|) \wedge \psi(x_1, \ldots, x_k, y)\Big].$$

The derived class $\forall\mathcal{C}$ is defined analogously with $\forall$ replacing $\exists$. The *polynomial hierarchy* PH of Meyer and Stockmeyer is the class

$$\mathrm{PH} = \mathrm{P} \cup \exists\mathrm{P} \cup \forall\exists\mathrm{P} \cup \exists\forall\exists\mathrm{P} \cup \cdots \tag{3}$$

of predicates obtained from the base class P by finite quantification. (Note the analogy with the Kleene's arithmetic hierarchy from classical recursion theory.) At the base of the hierarchy we find the classes P and NP = $\exists$P that were encountered in the previous section. Toda [17] has shown the following.

Theorem 1 *Every predicate in* PH *is polynomial-time Turing reducible to a function in* #P.

In other words, the class #P essentially contains the entire polynomial hierarchy. The power of the counting operator # to simulate arbitrary finite alternations of the operators $\exists$ and $\forall$ is, I think, surprising.

To give the flavour of the proof of Theorem 1, which relies on a subtle "quantifier swapping" argument, it is necessary to define two further operators of independent interest, $\oplus$ and B. A k-place predicate ϕ is contained in $\oplus\mathcal{C}$ if there is a $(k+1)$-place predicate $\psi \in \mathcal{C}$ and a polynomial p such that the number of words $y \in \Sigma^*$ of length $p(|x_1|, \ldots, |x_k|)$ satisfying $\psi(x_1, \ldots, x_k, y)$ is odd. A k-place predicate ϕ is contained in $\mathsf{B}\mathcal{C}$ if there is a $(k+1)$-place predicate $\psi \in \mathcal{C}$, a polynomial p, and a number $\alpha > 0$, such that $\Pr\big(\phi(x_1, \ldots, x_k) \Leftrightarrow \psi(x_1, \ldots, x_k, y)\big) \geq \frac{1}{2} + \alpha$, where $y \in \Sigma^*$ is selected uniformly at random (u.a.r.) from words of length $p(|x_1|, \ldots, |x_k|)$.

Toda showed that for predicate classes $\mathcal{C}$ satisfying a certain technical condition, which is satisfied by the classes forming the polynomial hierarchy (3),

$$\text{(i) } \exists\mathcal{C} \subseteq \mathsf{B}\oplus\mathcal{C}, \quad \text{(ii) } \oplus\mathsf{B}\oplus\mathrm{P} \subseteq \mathsf{B}\oplus\mathrm{P}, \quad \text{and} \quad \text{(iii) } \mathsf{B}\mathsf{B}\oplus\mathrm{P} \subseteq \mathsf{B}\oplus\mathrm{P}.$$

Inclusions (i) and (ii) are non-trivial, decidedly so in the case of (i), which rests on a universal hashing technique of Valiant and Vazirani [20]. If $\mathcal{C} \subseteq \mathsf{B}\oplus\mathrm{P}$ satisfies the technical condition, then

$$\exists\mathcal{C} \subseteq \mathsf{B}\oplus\mathcal{C} \subseteq \mathsf{B}\oplus\mathsf{B}\oplus\mathrm{P} \subseteq \mathsf{B}\mathsf{B}\oplus\mathrm{P} \subseteq \mathsf{B}\oplus\mathrm{P},$$

using containments (i)–(iii). Thus, by induction on quantifier depth, $\mathrm{PH} \subseteq \mathsf{B}\oplus\mathrm{P}$. The proof is completed by demonstrating that every predicate in the class $\mathsf{B}\oplus\mathrm{P}$ is polynomial-time Turing reducible to a function in #P.

3 Random instances

Under the assumption $\mathrm{P} \neq \mathrm{NP}$, there can be no polynomial-time algorithm for deciding Hamiltonicity or any other NP-complete predicate. This is a "worst-case" result: for any NP-complete predicate ϕ, and algorithm $\mathcal{A}$ for deciding ϕ, there exist instances $x \in \Sigma^*$ that cause $\mathcal{A}$ to run for superpolynomially many steps. However, these difficult instances may be sparse, so there is no reason to suppose that $\mathcal{A}$ cannot decide $\phi(x)$ quickly on most inputs. The same remarks hold good for #P-complete functions.

Empirically, most NP-complete predicates $\phi(x)$ seem easy to decide when the instance x is chosen at random, provided the probability distribution on instances is fairly natural. For example, deciding Hamiltonicity of a random graph seems in practice to be easy, whichever of the established random graph models is used; indeed, good results may be obtained using fairly simple heuristics.[2] In contrast, again from an empirical standpoint, counting seems to be hard for random instances. Suppose, for example, we are presented with an n-vertex graph G and asked to compute the number of Hamilton cycles in G. Whether the graph G is provided by an adversary or is selected u.a.r. seems to make no difference to the computational difficulty of the task. We can gain an understanding of why this is so through considering a phenomenon that has become known as "random self reducibility." Following Lipton [13], we find it convenient to investigate this phenomenon in the context of the permanent function.

Let $A = (a_{ij} : 0 \leq i, j \leq n-1)$ be an $n \times n$ matrix with entries in some ring. The *permanent* of A is defined as

$$\operatorname{per} A = \sum_{\pi} \prod_{i=0}^{n-1} a_{i,\pi(i)}\,,$$

[2] This rather sweeping statement deserves more substantial qualification than space here allows. Suffice it to say that the task of exhibiting algorithms that are provably efficient for almost every instance presents a considerable challenge.

where the sum is over all permutations π of $\{0, \ldots, n-1\}$. As noted in Section 1, the permanent is a #P-complete function, even when restricted to 0,1-matrices. Our aim is to show that evaluating the permanent of a random matrix R is as hard as evaluating the permanent of an arbitrary matrix. The first notion of "random matrix" that comes to mind is that of a random 0,1-matrix, since $\operatorname{per} R$ then has a natural combinatorial interpretation, namely, the number of perfect matchings in the bipartite graph with adjacency matrix R. However, the argument we present requires that the matrix elements take on a wider range of values, and we accordingly work with matrices over the finite field $\mathrm{GF}(p)$, where p is a prime not less than $n+2$.

Let A be an arbitrary $n \times n$ matrix over $\mathrm{GF}(p)$, which we may imagine to be chosen by some adversary. Our aim is to reduce the computation of $\operatorname{per} A$ to the computation of the permanents of a number of random matrices. Choose a random $n \times n$ matrix R, i.e, one with each entry selected independently, and u.a.r. from $\mathrm{GF}(p)$. The function $f(z) = \operatorname{per}(A + Rz)$ is a polynomial of degree n in the indeterminate z. If we knew the $n+1$ values $f(1), f(2), \ldots, f(n+1)$ then we could interpolate to find $f(0) = \operatorname{per} A$, which is precisely the quantity we wish to compute. But $f(i) = \operatorname{per}(A + iR)$, and $A + iR$ is a random matrix, provided $i \not\equiv 0 \pmod{p}$. It follows that the existence of a polynomial-time algorithm that correctly computes $\operatorname{per} A$ for all but a fraction $1/3(n+1)$ of $n \times n$ matrices A over $\mathrm{GF}(p)$ would imply the existence of a randomised polynomial-time algorithm that computes the permanent correctly, with high probability, on all inputs.

Using the theory of error correcting codes, it is possible to interpolate polynomials reliably from "noisy" data. This fact — which is implicit in a decoding procedure due to Berlekamp and Welch — allows the fraction $1/3(n+1)$ mentioned above to be increased to $\frac{1}{2} - \delta$ for any $\delta > 0$, which is clearly as much as can be achieved using a simple-minded application of random self reducibility. Using more sophisticated techniques, Feige and Lund [5] show that the existence of a polynomial-time algorithm that correctly computes the permanent of even a tiny proportion of all matrices would have a surprising consequence: that the polynomial hierarchy collapses to the second level. In the notation introduced in Section 2:

Theorem 2 *If there exists a polynomial-time algorithm that correctly computes the permanent on a fraction $100n^3/\sqrt{p}$ of all $n \times n$ matrices over $\mathrm{GF}(p)$, then $\exists\forall\mathrm{P} = \mathrm{PH}$.*

Note, however, that to obtain a dramatic effect it is necessary to work over rather large finite fields. Their proof exploits ideas from the theory of "interactive proofs."

Existing results concerning hardness of counting problems on random instances rely on polynomial interpolation, which can only work over a sufficiently large field. Intuitively, evaluating the permanent of a random 0,1-matrix ought to be as hard as evaluating the permanent of an arbitrary 0,1-matrix, and the same ought to be true of counting Hamilton cycles in a random graph, or of counting many other combinatorial structures in random graphs. (Note that these problems are "essentially combinatorial," in contrast to that of permanent evaluation over a finite field, which has an algebraic flavour.) There is no complexity-theoretic evidence that the cited problems are hard with respect to the uniform probability distribution on instances, and remedying this deficiency would be a significant advance.

4 Approximation algorithms

The familiar dichotomy within the class NP, namely P versus NP-complete, has a counterpart in the complexity class #P: almost all the "natural" functions in #P that have been considered have been shown either to be polynomial-time computable or to be #P-complete. Unfortunately, very few counting problems fall into the former category, notable examples being (a) spanning trees in a graph, which can be counted via Kirchhoff's matrix-tree theorem, (b) perfect matchings in a planar graph (including dimer coverings of a two dimensional lattice as a special case), the solution of which is a classical result of Kasteleyn [12], and (c) a few other structures — such as Eulerian circuits in a directed graph or Ising configurations in a planar lattice — that can be handled by fairly direct reduction to either (a) or (b). Valiant has made the intriguing observation that all the cited algorithms ultimately rely on linear algebra, in particular, the well known fact that the determinant of a matrix may be computed in polynomial time by Gaussian elimination.

Given the paucity of positive results, it is tempting to weaken the requirements somewhat, and consider whether *approximate* counting is feasible; for example, whether it is possible, in polynomial time, to evaluate the permanent of a 0,1-matrix with relative error at most 1%. The reader who has followed the recent dramatic advances in the field of approximation algorithms for combinatorial optimisation will be aware of the emerging classification of optimisation problems according to degrees of approximability: problems, such as chromatic number of an n-vertex graph, that cannot be approximated within ratio n^α for some $\alpha > 0$ unless P = NP; problems, such as "MaxSat," that can be approximated within ratio α but not within ratio β, for some $0 < \alpha < \beta < 1$, and so on. In contrast, Sinclair and the author [16] have demonstrated that no correspondly rich classification can be expected in the domain of approximate counting. A witness-checking predicate ψ is (downward) self reducible if, roughly, the set $\{y : \psi(x, y)\}$ has a simple recursive expresssion in terms of some collection of sets $\{y : \psi(x_1, y)\}, \ldots, \{y : \psi(x_k, y)\}$ with $|x_i| < |x|$ for all $1 \leq i \leq k$. Almost all naturally occurring predicates are self reducible in this sense.

Theorem 3 *Suppose ψ is a self reducible predicate, and let A be the set of $\alpha \in \mathbb{R}$ for which there exist $\delta > 0$ and a polynomial-time randomised algorithm that approximates the function $f(x) = \#^p y\, \psi(x, y)$ within ratio $1+|x|^\alpha$, with probability $\frac{1}{2} + \delta$. Then A is either empty or the whole of $\mathbb{R}$.*

In the absence of differing degrees of approximability, a single notion of efficient approximation algorithm has assumed considerable importance. Suppose $f : \Sigma^* \to \mathbb{N}$ is a function mapping problem instances to natural numbers. A *randomised approximation scheme* for f is a randomised algorithm that takes as input a word (instance) $x \in \Sigma^*$ and $\varepsilon > 0$, and produces as output a number Y (a random variable) such that $\Pr\Big((1-\varepsilon)f(x) \le Y \le (1+\varepsilon)f(x)\Big) \ge \frac{1}{2} + \delta$, for some $\delta > 0$.[3] A randomised approximation scheme is said to be *fully polynomial* if it runs in time polynomial in $|x|$ and ε^{-1}. The rather unwieldy phrase "fully polynomial randomised approximation scheme" is often abbreviated to *fpras*.

The above definition is due to Karp and Luby, who presented a simple but elegant fpras for counting satisfying assignments to a Boolean formula in DNF, a #P-complete problem. Unfortunately, the classical Monte Carlo approach adopted by Karp and Luby appears to have only limited application in this area. An alternative approach that has proved very fruitful exploits the close relationship that exists between approximate counting and (almost) uniform sampling. Subject to self reducibility, the existence of an efficient (i.e., polynomial-time) almost uniform sampling procedure for a certain combinatorial structure entails the existence of an efficient approximate counting procedure (i.e., fpras) for that structure, and vice versa. This connection was investigated by Valiant, Vazirani, and the author [10], and further elucidated in [16].

For the sake of definiteness, let us suppose that the goal is to estimate the permanent of a 0,1-matrix. We have noted that this task is equivalent to estimating the number of perfect matchings in a bipartite graph G, and hence to sampling perfect matchings in G almost uniformly. Broder [3] suggested the following approach to sampling matchings. Let M denote the set of all perfect matchings in G and M^- the set of "near-perfect" matchings that leave exactly two vertices uncovered. Construct a Markov chain (X_t) whose state space $\Omega = M \cup M^-$ consists of all perfect and near-perfect matchings. A transition from state $X_t \in \Omega$ to $X_{t+1} \in \Omega$ is possible if $X_t \oplus X_{t+1}$ — the symmetric difference of X_t and X_{t+1} — contains at most two edges; all allowed transitions occur with equal probability.

It can be shown that the Markov chain (X_t) is ergodic and has uniform stationary distribution. This observation suggests the following sampling procedure for perfect matchings: simulate the Markov chain from a fixed initial state X_0 for T steps; return as result the final state X_T if $X_T \in M$, otherwise repeat the simulation. If T is chosen large enough, the result of this procedure is a perfect matching chosen almost u.a.r. For this "Markov chain Monte Carlo" sampling procedure to be efficient, two conditions must be satisfied: (a) the ratio $|M|/|\Omega|$ must be non-negligible, and (b) the Markov chain (X_t) must be "rapidly mixing," that is to say, must be close to stationarity after a number of steps polynomial in n, the size of G. It is in verifying condition (b) that the work lies.

The classical approach to establishing that a Markov chain is rapidly mixing

[3] Any success probability strictly greater than $\frac{1}{2}$ may be boosted to a value arbitrarily close to 1 by making a small number of trials and taking the median of the results [10].

is via a "coupling argument." Broder attempted to use a coupling argument to prove that his perfect matching Markov chain is rapidly mixing; however, the technique appears ill-suited to the irregular Markov chains which typically arise in this context. A modest exception to this general rule is an algorithm of the author's for estimating the number of colourings of a low-degree graph [7], the correctness of which rests on a surprisingly simple coupling argument.

The failure of coupling to produce results has spurred the development of two new approaches to analysing rates of convergence of ergodic Markov chains. The first of these is the "canonical paths" argument, which was introduced by Sinclair and the author, and used by them to analyse Broder's perfect matching chain [8]. The canonical paths argument provides a lower bound on a quantity known as the *conductance* of the Markov chain, which is a weighted version of the familiar notion of expansion of a graph. Sinclair and the author [16] showed that conductance and mixing time (roughly, number of steps until the distribution on states is close to stationarity) are related, so that large conductance entails a short mixing time; similar results were obtained independently by Alon and Aldous, and can be regarded as discrete analogues of Cheeger's inequality.

The idea is to specify, for every pair of states $I, F \in \Omega$, a *canonical path*, starting at I and ending at F, and composed of valid transitions in the Markov chain. If the canonical paths can be chosen so that the total number of paths using any single transition is relatively small, then the conductance of the Markov chain must be relatively large, and the mixing time correspondingly short. This line of argument has been used to demonstrate the existence of an fpras for the permanent of a wide class of 0,1-matrices, including all matrices that are sufficiently dense [8], and for the partition function of a ferromagnetic Ising system [9].

The second approach is geometric in nature. The idea is to assign a geometric interpretation to the state space Ω, from which a lower bound on conductance can be read off using (enhancements of) classical isoperimetric inequalities. This was the approach used by Dyer, Frieze and Kannan [4] in their seminal work on estimating the volume of a convex body in high-dimensional Euclidean space, which has been variously refined by the original authors and by Lovász and Simonovits [14], and extended to the integration of log-concave functions over convex regions by Applegate and Kannan [1]. The approach was also used by Karzanov and Khachiyan [11] to validate an fpras for the number of linear extensions of a partial order.

It would be interesting to know which #P-complete functions can be approximated in the fpras sense, and which cannot. As we have seen, definite progress has been made in the last few years, at least on the positive side of the classification. However, many natural #P-complete functions have so far resisted analysis. There seem to be difficulties on both sides. On the one hand, several problems seem amenable to the Markov chain Monte Carlo method, but the Markov chains that have been proposed for their solution are not provably rapidly mixing. Further progress may be predicated on the development of new techniques for bounding the mixing-time of irregular Markov chains.

On the other hand, negative results are hard to come by, even though there are no doubt many natural #P-complete problems for which no fpras exists. Current techniques for establishing negative results are primitive. It is easy to show that if deciding existence of some combinatorial structure is NP-complete, then there can be no fpras for the number of occurrences of that structure unless RP = NP.[4] Furthermore, a simple argument [15, Thm 1.17] suffices to rule out the existence of an fpras for certain problems — including counting independent sets of all sizes in graph — again unless RP = NP. However, the current techniques seem to fall well short of resolving whether (for example) there is an fpras for the permanent of an *arbitrary* 0,1-matrix.

Acknowledgements I thank Sigal Ar, Alistair Sinclair, and Leslie Valiant for comments on a preliminary draft of this article.

References

[1] David Applegate and Ravi Kannan, *Sampling and integration of near log-concave functions*, Proceedings of the 23rd ACM Symposium on Theory of Computing, ACM Press, 1991, 156–163.

[2] Graham Brightwell and Peter Winkler, *Counting linear extensions*, Order **8** (1991), 225–242.

[3] Andrei Z. Broder, *How hard is it to marry at random? (On the approximation of the permanent)*, Proceedings of the 18th ACM Symposium on Theory of Computing, ACM Press, 1986, 50–58. Erratum in Proceedings of the 20th ACM Symposium on Theory of Computing, 1988.

[4] M. Dyer, A. Frieze, and R. Kannan, *A random polynomial time algorithm for approximating the volume of convex bodies*, Journal of the ACM **38** (1991), 1–17.

[5] Uriel Feige and Carsten Lund, *On the hardness of computing the permanent of random matrices*, Proceedings of the 24th ACM Symposium on Theory of Computing, ACM Press, 1992, 643–654.

[6] F. Jaeger, D. L. Vertigan, and D. J. A. Welsh, *On the computational complexity of the Jones and Tutte polynomials*, Mathematical Proceedings of the Cambridge Philosophical Society **108** (1990), 35–53.

[4] RP is is the class of predicates that can be decided by a polynomial-time *randomised* algorithm.

[7] Mark Jerrum, *A very simple algorithm for estimating the number of k-colourings of a low-degree graph*, Report ECS-LFCS-94-290, Department of Computer Science, University of Edinburgh, April 1994.

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

[9] Mark Jerrum and Alistair Sinclair, *Polynomial-time approximation algorithms for the Ising model*, SIAM Journal on Computing **22** (1993), 1087–1116.

[10] Mark R. Jerrum, Leslie G. Valiant and Vijay V. Vazirani, *Random generation of combinatorial structures from a uniform distribution*, Theoretical Computer Science **43** (1986), 169–188.

[11] A. Karzanov and L. Khachiyan, *On the conductance of order Markov chains*, Technical Report DCS 268, Rutgers University, June 1990.

[12] P. W. Kasteleyn, *Dimer statistics and phase transitions*, Journal of Mathematical Physics **4** (1963), 287–293.

[13] Richard J. Lipton, *New directions in testing*, Distributed Computing and Cryptography (J. Feigenbaum and M. Merritt, eds), DIMACS Series in Discrete Mathematics and Theoretical Computer Science **2**, American Mathematical Society, Providence RI, 1991, 191–202.

[14] L. Lovász and M. Simonovits, *Random walks in a convex body and an improved volume algorithm*, Random Structures and Algorithms (to appear).

[15] Alistair Sinclair, *Algorithms for random generation and counting: a Markov chain approach*, Progress in Theoretical Computer Science, Birkhäuser, Boston, 1993.

[16] Alistair Sinclair and Mark Jerrum, *Approximate counting, uniform generation and rapidly mixing Markov chains*, Information and Computation **82** (1989), 93–133.

[17] Seinosuke Toda, *On the computational power of* PP *and* $\oplus$P, Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science, Computer Society Press (1989), 514–519.

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

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

[20] L. G. Valiant and V. V. Vazirani, NP *is as easy as detecting unique solutions*, Theoretical Computer Science **47** (1986), 85–93.