SIAM J. COMPUT.
Vol. 8, No. 3, August 1979

THE COMPLEXITY OF ENUMERATION AND RELIABILITY PROBLEMS*

LESLIE G. VALIANT†

Abstract. The class of #*P*-complete problems is a class of computationally eqivalent counting problems (defined by the author in a previous paper) that are at least as difficult as the *NP*-complete problems. Here we show, for a large number of natural counting problems for which there was no previous indication of intractability, that they belong to this class. The technique used is that of polynomial time reduction with oracles via translations that are of algebraic or arithmetic nature.

Key words. counting, enumeration, reliability, computational complexity, *NP*-completeness, permanent, matchings

1. Introduction. It is an empirical fact that for numerous combinatorial problems the detection of the existence of a solution is easy, yet no computationally efficient method is known for counting their number. The purpose of this paper is to show that for a variety of well-known problems this phenomenon can be explained. We define the class of #*P*-complete functions as in [20]. Typical members of this class are the problems of counting the number of solutions of *NP*-complete problems. We show that for many natural structures that are apparently unrelated to any *NP*-complete structure, the problem of counting them is nevertheless #*P*-complete. The notion of reducibility used is that of polynomial time transduction with oracles. The reductions themselves are characterized by being of an algebraic or arithmetic, rather than combinatorial nature.

The more significant problems that are shown to be #*P*-complete are: counting perfect matchings in bipartite graphs [20]; counting trees in a directed graph; counting satisfying assignments to monotone Boolean formulae in 2-conjunctive normal form; counting maximal cliques (i.e. nonextendable complete subgraphs); and evaluating the probability that two given nodes in a probabilistic network are connected.

Many apparently difficult counting problems are probably not candidates for being #*P*-complete. Among these are questions of the form: how many graphs of size n are there that have property X? Since for each n there is just one input, these problems correspond to *NP* computations over a single-letter input alphabet. We call this class $\#P_1$ and exhibit a natural problem that is complete in it. A variant of the problem has the additional curious property that while it is provably as complex as any $\#P_1$-complete problem, it is not necessarily complete itself.

The completeness results have a direct bearing on the classical study of enumerations. Note, however, that the notion of "effectively counting" that we use here is that of polynomial time computability. Since discrete probabilistic problems can usually be reformulated as counting problems, our techniques can also be applied to reliability problems, of which connectedness is a typical example. A third field of application is to "branch and bound" or search algorithms. Some simple examples of these essentially enumerate some easily detectable structure (e.g. maximal cliques). Our results suggest potential techniques for proving for such algorithms that the problem of predicting from an input the runtime of the algorithm on that input is #*P*-complete.

2. Preliminaries. In the main we use the definitions introduced in [20]. A more general schema of definitions and some discussion of them can be found there. For background on *NP*-completeness see [1], [5], [10].

* Received by the editors November 18, 1977.

† Department of Computer Science, University of Edinburgh, Edinburgh, Scotland.

DEFINITION. A *counting Turing machine* is a standard nondeterministic TM with an auxiliary output device that (magically) prints in binary notation on a special tape the number of accepting computations induced by the input. It has (worst-case) *time-complexity* $f(n)$ if the longest accepting computation induced by the set of all inputs of size n takes $f(n)$ steps (when the TM is regarded as a standard nondeterministic machine with no auxiliary device).

DEFINITION. $\#P$ is the class of functions that can be computed by counting TMs of polynomial time complexity. $\#P_1$ is defined similarly for TMs with a unary input alphabet.

We denote the class of functions computed by deterministic polynomial time TMs by *FP*, and the class of predicates by *P*. For convenience we shall often identify a class of machines with the class of functions it computes. It will be assumed that objects are represented in some standard economical manner as words over an alphabet Σ (say $\{0, 1\}$). $|x|$ will denote the size of x if x is a set, and its length if x is a string. A function $f: \Sigma^* \to \Sigma^*$ (or a relation $R \subseteq \Sigma^* \times \Sigma^*$) is *polynomial bounded* iff there is a polynomial p such that for all x, $|f(x)| < p(|x|)$ (or such that $R(x, y) \Rightarrow |y| < p(|x|)$).

The notion of reduction used is one by oracles, in a similar sense to Cook [5] except that the oracles cannot only be predicates but also arbitrary polynomial bounded functions. An *oracle TM* is a TM with a query tape, an answer tape, and some working tapes. To consult the oracle the TM prints a word on the query tape, it goes into a special query state and returns an answer in unit time on the answer tape, and it enters a special answer state. An oracle TM is said to be in *P* (or *FP*, or *NP*, or $\#P$, etc.) iff for all polynomial bounded oracles it behaves like a machine in *P* (or *FP*, or *NP* or $\#P$, etc.).

If α is a class of oracle-TMs and x an appropriate function for it (i.e. polynomial bounded in the present context) then we denote the class of functions that can be computed by oracle-TMs from α with oracles for x by α^x. The class of functions that can be computed by just a single call of the oracle for any input is denoted by $\alpha^{x!}$. A problem y is *$\#P$-hard* iff $\#P \subseteq FP^y$. It is *$\#P$-complete* iff $\#P \subseteq FP^y$ and $y \in \#P$. In expressing reductions between two problems it is useful to abbreviate $x \in FP^y$ by $x \leqq y$ and $x \in FP^{y!}$ by $x \leqq ! y$. Notice that both binary relations are transitive.

A relation R is *P-enumerable* iff there is a polynomial p such that for all x the set $\{y | R(x, y)\}$ can be enumerated in time $|\{y | R(x, y)\}| \cdot p(|x|)$.

3. Lemmas.

Let SAT be the problem of counting the number of satisfying assignments of a Boolean formula F in conjunctive normal form, and let 3-SAT be the same problem for formulae with at most three disjuncts in each conjunct. Let TM-COMP be the problem of counting the number of accepting computations given an arbitrary polynomial time nondeterministic TM and an input for it. HAMILTONIAN CIRCUITS is the problem of counting the number of such circuits in a graph. (N.B. Here as elsewhere in the paper, graphs can be interpreted either as being directed or as being undirected, unless otherwise indicated.)

FACT 1. TM-COMP $\leqq !$ SAT.

Proof. The transformation of Cook as given in [5] or [1] establishes this. □

FACT 2. SAT $\leqq !$ 3SAT.

Proof. Suppose F has a clause with $i \geqq 4$ literals. If we replace in the clause any two of these literals (e.g. x_j, $\bar{x}_k$) by a new variable (say y) and conjoin F with the CNF formula for $(x_j \vee \bar{x}_k) \equiv y$ (which has clearly at most 3 literals per clause) then the new formula will have the same number of solutions as F. The result follows by induction. □

FACT. 3. SAT $\leqq !$ HAMILTONIAN CIRCUITS.

Proof. A direct reduction that preserves the number of solutions is given in [19] for both the directed and undirected cases. □

FACT 4. *Given an $n \times n$ integer matrix with each entry bounded in magnitude by 2^m the determinant and inverse can be computed in time polynomial in n and m.*

Proof. Perform Gaussian elimination with arithmetic modulo 2^k, the smallest power of 2 greater than $2^{mn} \cdot n!$. As pivot always choose a number that has the fewest factors of 2 in its prime decomposition. This ensures that eliminating with respect to that row will multiply the matrix by various numbers coprime with 2^k. When an upper diagonal matrix is achieved the value of the determinant can be computed by dividing the products of the diagonal elements by the product of the multipliers.

The inverse can be computed as a rational number by the determinental rule for example. □

In the remaining two facts the *size* of a rational number will be the sum of the lengths of its numerator and denominator (assumed coprime).

FACT 5. (i) *If $p(x)$ is an n-th degree polynomial and its value is known at each of the rational points $x_1, \cdots, x_{n+1}$, all of size at most m, then the coefficients of p can be deduced in time polynomial in n, m and the size of the largest value.*

(ii) *If the value of*

$$p(x, y) = \sum_{i=1}^{q} \sum_{j=1}^{r} p_{ij} x^i y^j$$

is known for all pairs of points $x = x_h$, $y = y_k$ for $1 \leqq h \leqq q+1$ and $1 \leqq k \leqq r+1$ (all points being of size bounded by m) then the value of each p_{ij} can be deduced in time polynomial in q, r, m and the size of the largest value.

Proof. (i) Let X be the $(n+1) \times (n+1)$ matrix with $X_{ij} = x_i^{j-1}$. Then X is Vandermonde and has an inverse, which, by Fact 4, can be computed fast. But if $\mathbf{p}$ is the vector of coefficients of p and $p(\mathbf{x})$ the vector of values at the $n+1$ points then $p(\mathbf{x}) = X\mathbf{p}$ and hence $\mathbf{p} = X^{-1} p(\mathbf{x})$.

(ii) For each value x_h use (i) to compute the value of $\sum p_{ij} x_h^i$ for each j. Then use (i) again to deduce each p_{ij}. □

FACT 6. *Let $\{a_i\}$ and $\{b_{ij}\}$ be sets of positive integers bounded by $A > 2$. If the value of any one of the following functions is known at a suitable point x_0, or (x_0, y_0), then the value of each a_i, or each b_{ij} can be deduced in time polynomial in n, m and in the sizes of x_0, y_0 and the value.*

(i) $\sum_0^n a_i x^i$ *if* $x_0 \geqq A^2$ *or* $0 < x_0 \leqq A^{-2}$,

(ii) $\sum_0^n a_i x^i (1-x)^{n-i}$ *if* $0 < x_0 \leqq A^{-2}$,

(iii) $\sum_{i=0}^{n} \sum_{j=0}^{m} b_{ij} x^i (1-x)^{n-i} y^j (1-y)^{m-j}$ *if* $x_0 = A^{-2}$ *and* $O < y_0 < A^{-3n}$.

Proof. (i) If $x_0 \geqq A^2$ then for each j

$$\sum_{i=0}^{j-1} a_i x_0^i < A(x_0^{j-1}(1 + A^{-2} + A^{-4} + \cdots))$$
$$< 3A \cdot x_0^{j-1}/2.$$

Hence $x_0^j > \sum_0^{j-1} a_i x_0^i$. It follows that $a_n, a_{n-1}, \cdots, a_0$ can be computed in succession.

The alternative case follows similarly by examining the reciprocal of x_0.

(ii) Let $t = 1/x_0$, so that $t > A^2$. Then

$$\sum_{j+1}^{n} a_i x_0^i (1-x_0)^{n-i} = t^{-n} \cdot \sum_{j+1}^{n} a_i (t-1)^{n-i}$$
$$< 3t^{-n} \cdot A(t-1)^{n-j}/2.$$

Hence $(t-1)^{n-j} > \sum_{j+1}^{n} a_i(t-1)^{n-i}$, and $a_0, a_1, \cdots, a_n$ can be computed in succession.

(iii) The substitution for x_0 gives the value

$$\sum_j \sum_i b_{ij}(A^2-1)^{n-i} y^j (1-y)^{m-j}.$$

Applying the argument of (ii) to the outer summation gives $\sum b_{ij}(A^2-1)^{n-i}$. Applying it again gives the coefficients. □

FACT 7. *Suppose $R(x, y)$ is a polynomial bounded relation and the sets $R = \{(x, y)|R(x, y)\}$ and $R^1 = \{(x, \alpha)|\exists \beta \in \{0, 1\}^*$ s.t. $R(x, \alpha\beta)\}$ are both polynomial time recognizable. Then R is P-enumerable.*

Proof. A call, Enumerate $(x, \emptyset)$, of the following recursive procedure clearly suffices:

```
procedure Enumerate $(x, \alpha)$.
begin if $R(x, \alpha)$ then output $\alpha$;
        if $R^1(x, \alpha)$ then Enumerate $(x, \alpha 0)$ and Enumerate $(x, \alpha 1)$;
end. □
```

4. Some #*P*-complete problems. Unless otherwise stated we shall denote a graph, whether directed or undirected, by $G = (V, E)$ where $V = (u_1, \cdots, u_n)$ and $E \subseteq (V \times V)$. $G_1 = (V_1, E_1)$ is a *subgraph* of G if $V_1 = V$ and $E_1 \subseteq E$. By F we shall denote a Boolean formula in conjunctive normal form, with clauses $c_1, \cdots, c_r$ and variables $X = \{x_1, \cdots, x_n\}$. F is *monotone* if no variable is negated. It is in *k-form* if each conjunct has at most k disjuncts. $\mathbf{x}$ will denote an n-tuple from $\{0, 1\}^n$. $F(\mathbf{x})$ denotes the truth value of F when the ith component of $\mathbf{x}$ is substituted as the truth value of x_i.

We shall first specify a list of counting problems (2–14). As can be verified easily each one is in #*P*. Also, most of them are *P*-enumerable by virtue of Fact 7.

1. PERMANENT
   Input: Integer matrix A.
   Output: $\sum_\sigma \prod_{i=1}^n A_{i,\sigma(i)}$ summed over all permutations σ on $\{1, \cdots, n\}$.
2. PERFECT MATCHINGS
   Input: Bipartite graph G with $2n$ nodes.
   Output: Number of perfect matchings (i.e. sets of n edges such that no pair of edges has a common node).
3. MONOTONE PRIME IMPLICANTS
   Input: Monotone F in 2-form.
   Output: $|\{Y \subseteq X|(\bigwedge_{x \in Z} x \Rightarrow F)$ holds for $Z = Y$ but not for any $Z \subsetneq Y\}|$.
4. MINIMAL VERTEX COVER
   Input: G.
   Output: $|\{V' \subseteq V|$“$(u, v) \in E \Rightarrow u \in A$ or $v \in A$” holds for $A = V'$ but not for any $A \subsetneq V'\}|$.
5. MAXIMAL CLIQUES
   Input: G
   Output: $|\{V' \subseteq V|$“$u, v \in A \Rightarrow (u, v) \in E$” holds for $A = V'$ but not for any $A \supsetneq V'\}|$.

6. IMPERFECT MATCHINGS
   Input: Bipartite graph with $2n$ nodes.
   Output: Number of matchings of any size.
7. MONOTONE 2-SAT
   Input: $F = c_1 \wedge c_2 \wedge \cdots \wedge c_r$ where $c_i = (y_{i1} \vee y_{i2})$ and $y_{ij} \in X$.
   Output: $|\{\mathbf{x} | F(\mathbf{x}) \text{ true}\}|$.
8. SAT′
   Input: As for 7.
   Output: $|\{(\mathbf{x}, \mathbf{t}) | \mathbf{t} = (t_1, \cdots, t_n) \in \{1, 2\}^n;$ for $1 \leqq i \leqq r$, $\mathbf{x}$ makes $y_{i,k}$ true for $k = t_i\}|$.
9. SAT″
   Input: As in 7.
   Output: $|\{(\mathbf{x}, \mathbf{t}) | \mathbf{t} = (t_1, \cdots, t_n) \in \{\{1\}, \{2\}, \{1, 2\}\}^n;$ for $1 \leqq i \leqq r$ $\mathbf{x}$ makes $y_{i,k}$ true for each $k \in t_i\}|$.
10. S-SET CONNECTEDNESS (directed and undirected)
    Input: G; $s \in V$; $V' \subset V$.
    Output: Number of subgraphs of G in which for each $u \in V'$ there is a (directed) path from s to u.
11. S–T CONNECTEDNESS (directed or undirected)
    Input: G; $s, t \in V$.
    Output: Number of subgraphs of G in which there is a (directed) path from s to t.
12. S–T NODE CONNECTEDNESS (directed or undirected)
    Input: G; $s, t \in V$.
    Output: Number of subsets of V whose removal leaves a (directed) path from s to t.
13. DIRECTED TREES
    Input: Directed graph G.
    Output: Number of sets of edges that form a rooted tree, with each edge directed away from the root.
14. S–T PATHS (i.e. SELF-AVOIDING WALKS) (directed or undirected)
    Input: G; $s, t \in V$.
    Output: Number of (directed) paths from s to t that visit every node at most once.

We note that several of these problems have been widely studied. Because of the close resemblance between the permanent and the determinant the apparent computational discrepancy has been observed with surprise for a long time [16]. Despite considerable efforts no general translation from the former to the latter has been found [17], [14]. In special cases, however, such transformations do exist and lead to fast algorithms (G. Borchardt (1855), see [3]), [11], [13], [15], [18]. The maximal cliques problem arises in connection with the numerous algorithms that have been proposed for enumerating them [4], [9]. S–T paths are discussed in [2], [12]. Problems 10–12 are classical examples of reliability problems concerning networks, in the special case that the probability associated with each node or edge is a half. It will be clear, however, that the completeness results follow also for other fixed values (and of course for arbitrary values). Such problems are discussed in [6], [7]. The directed trees problem is to be contrasted with the directed spanning tree problem which can be counted fast via determinants, in analogy with Kirchhoff's matrix-tree theorem [8], [22].

THEOREM 1. *Problems* 2–14 *above are all #P-complete.*

Proof. The result follows by transitivity from the following reductions.

1. 3-SAT $\leqq$! PERMANENT. Proved in [20].
2. PERMANENT $\leqq$ PERFECT MATCHINGS. Proved in [20].
3. PERFECT MATCHINGS $\leqq$! PRIME IMPLICANTS. Given G with $V =$

$\{u_1, \cdots, u_n, v_1, \cdots, v_n\}$, $E \subset \{(u_i, v_j) | 1 \leqq i, j \leqq n\}$, we construct $G' = (V', E')$ with

$$V' = \{u_i^j, v_i^j | 1 \leqq i \leqq n; 1 \leqq j \leqq k = 2n\},$$

and

$$E' = \{(u_i^j, v_p^q) | (u_i, v_p) \in E; 1 \leqq j, q \leqq k\}.$$

We represent each edge of G' by a separate Boolean variable in which truth will denote the absence of the edge. The simultaneous presence of any pair of edges can then be prohibited by the disjunction of the corresponding variables. We can therefore write a polynomial length monotone formula F in 2-form that has the effect of prohibiting the presence of any pair of edges in G' that *either* arise from distinct edges of G that share a node *or* themselves share a node in G'. Thus to each matching of size i in G we intend there to correspond $(k!)^i$ allowed matchings in G'. Note that any prime implicant of F has $|E'| - ik$ literals for some i.

Now if $F(\mathbf{x})$ is true then at least $|E'| - nk$ of the $|E'|$ variables must be true (i.e. representing at most nk edges). If exactly $|E'| - nk$ are true then their conjunction must be a prime implicant, and corresponds to some perfect matching in G'. If I_i is the number of prime implicants of F with exactly $|E'| - ik$ literals and M_i the number of maximal matchings with i edges in G, then

$$I_i = M_i(k!)^i.$$

Hence if $\sum I_i$ is known then, by Fact 6(i), since $k! = (2n)! > M_i^2$, the value of M_n, the number of perfect matchings in G can be deduced.

4. PRIME IMPLICANTS$\leqq !$MINIMAL VERTEX COVER. Given F construct a $G = (V, E)$ with $V = \{u_1, \cdots, u_n\}$ and $E = \{(u_i, u_j) | (x_i \vee x_j)$ is a clause of $F\}$. Any vertex cover of G is an implicant of F, and any minimal vertex cover is a prime implicant.

5. MINIMAL VERTEX COVER$\leqq !$MAXIMAL CLIQUES. A minimal vertex cover in G corresponds to a maximal clique in the complement of G, in analogy with [10].

6. PERFECT MATCHINGS$\leqq$IMPERFECT MATCHINGS. Given $G = (V, E)$ as in 3 we construct for each k $(1 \leqq k \leqq n+1)$ a graph G_k that consists of G with additional nodes $\{u_{ij} | 1 \leqq i \leqq n; 1 \leqq j \leqq k\}$ and additional edges $\{(u_{ij}, v_i) | 1 \leqq i \leqq n; 1 \leqq j \leqq k\}$. If A_r is the number of matchings in G of size exactly $n - r$ then these will be contained in exactly $A_r \cdot (k+1)^r$ imperfect matchings obtainable by adding only new edges. Hence the number of matchings in G_k is $\sum_{r=0}^n A_r \cdot (k+1)^r$. If this could be evaluated for $k = 1, \cdots, n+1$, then, by Fact 5, we could compute A_0, the number of perfect matchings.

7. IMPERFECT MATCHINGS$\leqq !$MONOTONE 2-SAT. For G as defined in 3 above represent (the abence of) each edge by a separate variable. Let F be the formula that prohibits the presence of any pair of edges incident to the same node.

8. MONOTONE 2-SAT$\leqq !$SAT$'$. Given F, denote $F \wedge F \wedge \cdots \wedge F$, k times, by F^k. If $\mathbf{x}$ satisfies exactly f clauses of F twice and the rest once, then it contributes 2^f to SAT$'$ for F and 2^{kf} for F^k. If A_f is the number of assignments that satisfy exactly f clauses of F twice and the rest once, then the value of SAT$'$ for F^k is $\sum_{f=0}^r A_f(2^k)^f$. By choosing k suitably large it follows from Fact 6(i) that $\sum A_f$ can be deduced.

9. MONOTONE 2-SAT$\leqq !$SAT$''$. This is similar to 8.

10. SAT$'\leqq !S$-SET CONNECTEDNESS. Given F construct a graph $G =$

$(V, E_1 \cup E_2)$ where

$$V = \{c_1, \cdots, c_{r+1}, x_1, \cdots, x_n, \bar{x}_1, \cdots, \bar{x}_n, s\},$$

$$E_1 = \{(x_i, c_j) | x_i \text{ appears in clause } c_j \text{ in } F\} \cup \{(\mathrm{x}_n, \mathrm{c}_{r+1}), (\bar{\mathrm{x}}_n, \mathrm{c}_{r+1})\},$$

$$E_2 = \{(x_i, x_{i+1}), (\bar{x}_i, \bar{x}_{i+1}), (x_i, \bar{x}_{i+1}), (\bar{x}_i, x_{i+1}) | 1 \leqq i \leqq n\} \cup \{(s, x_1), (s, \bar{x}_1)\}.$$

Suppose each edge in E_1 is given probability p and each one in E_2 probability q. Let A_{ij} be the number of distinct subsets of $E_1 \cup E_2$ with exactly i edges from E_1 and j from E_2, such that s is connected to each of $c_1, c_2, \cdots, c_{r+1}$. Then the probability that s is connected to each of $c_1, \cdots, c_{r+1}$ is

$$\sum\sum A_{ij} p^i (1-p)^{2r+2-i} q^j (1-q)^{4n-2-j}.$$

By Fact 6(iii), if we can evaluate this for sufficiently small p and q (e.g. $p = 2^{-2m}$, $q = 2^{-3m^2}$ where $m = 4n + 2r$) then we can compute $\{A_{ij}\}$. Such probabilities as 2^{-2m} can be simulated simply by replacing the edge by a chain of $2m$ edges of probability $1/2$. The result follows since $A_{r+1,n}$ is the required solution to SAT′ for F. (N.B. The fact that SAT′ is defined for monotone 2-form, rather than general 3-form, is inessential to this proof.)

11. S-SET CONNECTEDNESS $\leqq$ S–T CONNECTEDNESS. Given $G = (V, E)$ and $V' = \{n_1, \cdots, n_k\}$ construct G' by adding to G a node t and edges $\{(n_i, t) | 1 \leqq i \leqq k\}$. Suppose A_i is the number of subgraphs of G in which s is connected to exactly i nodes from V'. If each edge incident to t is given probability $1-p$ and all the others probability a half then the probability that s is connected to t is $2^{-|E|} \cdot \sum A_i(1-p^i)$. It follows from Fact 5 that by evaluating this at $k+1$ points the value of A_k can be deduced. The points can be taken as $1-p = 2^{-i}$ for $i = 1, \cdots, k+1$ since chains of suitable length can simulate these probabilities.

12. SAT′ $\leqq$ S–T NODE CONNECTEDNESS.This is similar to 10 and 11 combined.

13. SAT′ $\leqq$ DIRECTED TREES. Given F we construct exactly the same G as in 11, with edges directed in the sense indicated by their definitions. Suppose G_{pq} is obtained from G by giving each edge in E_1 multiplicity p, and each in E_2 multiplicity q. Let A_{ij} be the number of trees of G rooted as s with i edges from E_1 and j edges from E_2. Then the number of trees in G_{pq} rooted as s is

$$\sum_{i=1}^{2r+2} \sum_{j=1}^{4n-2} A_{ij} p^i q^j.$$

Hence by Fact 5(ii), if this is evaluated for all pairs (p, q) with $1 \leqq p \leqq 2r+3$ and $1 \leqq q \leqq 4n-1$ then the value of any A_{ij} can be deduced, including $A_{r+1,n}$ which is the desired result for SAT′. Now note that G_{pq} can indeed be simulated by an ordinary graph G'. G' consists of G augmented by chains of length p starting from each c_i, and chains of length q starting at each x_i and each $\bar{x}_i$. Finally observe that if we could count trees rooted arbitrarily, then by doing this for G' and again for G' with s removed, we could count the number of trees rooted as s.

14. HAMILTONIAN CIRCUITS $\leqq$ S–T PATHS. Given G for $k = 1, \cdots, n+1$, we generate a graph G_k by replacing each edge (u_i, u_j) by the graph with nodes $\{u_i, u_j\} \cup \{u_{ij}^1, \cdots, u_{ij}^k\}$ and edges $\{(u_i, u_{ij}^q), (u_{ij}^q, u_j) | q = 1, \cdots, k\}$. Then each s–t paths of length p in G corresponds to k^p s–t paths in G_k. Hence if there are A_p s–t paths of length p in G then the number of s–t paths in $G_k = \sum_{p=0}^{n} A_p k^p$. From Fact 5 it follows that A_n can be deduced, which is the number of Hamiltonian paths from s to t. The correspondence between Hamiltonian paths and cycles is immediate. (N.B. There is also a

natural $\leqq !$ reduction consisting of replacing each edge in G by a graph with exponentially many paths through it. Note also that the case of $s = t$ corresponds to enumerating elementary cycles [22], [23].) $\square$

The reductions used above can be classified according to whether (a) there is just one oracle call, or (b) there are several, but all the questions for it are generated without calling the oracle. Note that reductions of the latter type can *always* be replaced by one for the former if the problem to which reduction is being exhibited is appropriate: e.g. SAT, MONOTONE 2-SAT, SAT′, S–T CONNECTEDNESS, S–T PATHS. These problems can all exploit Fact 6 by being able to simulate the necessary arithmetic. We illustrate this for MONOTONE 2-SAT: Given F, to multiply its solutions by a large constant 3^k, we simply introduce $2k$ new variables and k new clauses containing two each. To multiply F_1 and F_2 we ensure that they have disjoint alphabets and simply write $F_1 \wedge F_2$. For addition of F_1 and F_2 consider F the conjunction of F_1 and F_2 with

$$(x_i \vee z) \cdots (x_n \vee z)(z \vee t)(y_1 \vee t) \cdots (y_m \vee t)$$

where the x's and y's are the variables of F_1 and F_2 respectively. Then $s(F)$ the number of solutions will equal $s(F_1)+s(F_2)+s(F_1)s(F_2)$. Hence to add we first multiply by a constant larger than the addends and perform this construction.

A further problem that can be shown to be $\#P$-hard is that of counting the number of Hamiltonian subgraphs of an arbitrary directed graph. This problem, however, appears not to belong to $\#P$. Corresponding problems for other NP-complete structures can also be formulated. Some of them appear surprisingly difficult to analyze.

For certain counting problems in $\#P$ for which no polynomial time algorithm is known, it is possible to prove that they can be computed in polynomial time given an oracle for some predicate in the Meyer–Stockmeyer hierarchy. An example is the problem of counting graph isomorphisms. Such a result can be interpreted as circumstantial evidence that the problem is not $\#P$-complete [21].

5. A problem complete in $\#P_1$.

Given a complete graph on n nodes and arbitrary probabilities assigned to each edge, the probability that the graph has a Hamiltonian circuit (or some other NP-complete substructure) is easily seen to be $\#P$-complete. If, however, we insist on all the probabilities being equal to a half then the corresponding problems (i.e. of counting the number of Hamiltonian graphs, etc., of a given size) are all open. Many of the classical graph enumeration problems are of this form [8]. Here we shall give an illustrative example to show that some such problem is provably $\#P_1$-complete. We note that the arithmetic reduction needed here takes the form of the "inclusion-exclusion" principle.

We assume a fixed collection of *colours* each associated with a number. By a *graph* we shall here mean a "connected directed graph in which each edge and node is assigned a colour, with the restriction that the number of edges meeting at a node has to equal the number associated with the colour of the node". A *pattern* is such a graph without the latter degree restriction. A pattern G, can be *embedded* in graph G_2 iff there is an injective mapping of the nodes of G_1 into those of G_2 such that all nodes and edges map to corresponding colours, and edges preserve direction.

Let $A = \{A_1, \cdots, A_k\}$ be an arbitrary collection of patterns, and consider the following problem schemes:

A-PATTERNS

Input: Integer n in unary.

Output: Number of labelled graphs with n nodes into which A_i can be embedded for *all* i $(1 \leqq i \leqq k)$.

A-SUBSET-PATTERNS

Input: Integer n in unary; $X \subseteq \{1, \cdots, k\}$.

Output: Solution for A'-PATTERNS where A' is the subset of A indexed by X.

THEOREM 2. *There is a fixed collection B of fixed patterns such that* B-*SUBSET-PATTERNS is* $\#P_1$*-complete.*

COROLLARY. *There is a fixed collection A of fixed patterns such that A-PATTERNS* $\in FP \Leftrightarrow \#P_1 \subseteq FP$.

Proof. Set B must have a subset A (although we may not be able to identify it). $\square$

Proof of Theorem 2. The problem is obviously in $\#P_1$. To show that it is complete we show how an arbitrary counting multi-tape TM, M, over a single letter input alphabet can be simulated by it.

We first modify M so that all accepting computations on the input of size n run for exactly $S(n)$ steps where S is an easily computed polynomial. We do this by simulating on an extra tape a binary counter that counts up to some simple polynomial that exceeds the complexity of M. The runtime of such a counter (whatever the implementation details) will clearly have some fixed value $S(n)$ that is easily computed from n. In all computation branches the modified machine M' simply runs the counter and simulates M in parallel until the former terminates. M' accepts if and only if it has simulated an accepting state of M at some time in the computation. It will be convenient to assume from now on that M and M' work on semi-infinite tapes.

We next modify M' to M'' so that M'' has just one tape but retains the property that all accepting computations have the same easily predetermined runtime. M'' treats its tape as a multiple-track tape. It initially checks that the input is of the form $1^n b^{S(n)-n-1}\$$ (where b and \$ are special new symbols) and rejects otherwise. These $S(n)$ squares are designated as work-space. In each step of the simulation of M' the workspace is scanned in both directions so as to take a fixed amount of time.

Let M'' have time complexity $T(n)$ and space complexity $S(n)$. We now claim that there is a set $C = \{C_1, \cdots, C_i\}$ of compulsory graphs, and a set $F = \{F_1, \cdots, F_j\}$ of forbidden graphs such that the number of accepting computations of M'' on input n is just the number of $(T(n)+1)(S(n)+1)$-node graphs in which all the compulsory graphs can be embedded but none of the forbidden ones. If we denote the number of graphs of this size that contain all of $C' \subseteq C \cup F$ and none of $F' \subseteq F$ as embedded subgraphs by $X(C', F')$, then clearly

$$X(\{C_1, \cdots, C_i\}, \{F_1, \cdots, F_j\}) = X(\{C_1, \cdots, C_i\}, \{F_2, \cdots, F_j\}) - X(\{C_1, \cdots, C_i, F_1\}, \{F_2, \cdots, F_j\}).$$

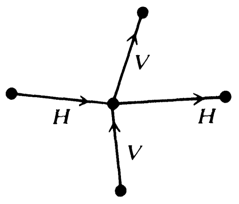
It follows by induction that if we can compute $X(\{A\}, \emptyset)$ for all $A \subseteq C \cup F$ then we can compute $X(C, F)$. The theorem therefore follows.

To prove the claim we represent computations by connected rectangular grids as shown in Fig. 1. Each horizontal line encodes a tape symbol and information about whether the head is there and, if so, the state of the machine. We ensure that only rectangular grids are counted by means of the following:

(i) Horizontal and vertical lines have disjoint colour sets H and V.

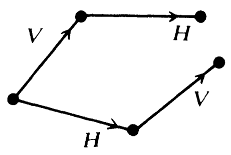
(ii) Nine distinct colours are used to distinguish from each other nodes that are on the four corners, on the four sides and internal, respectively. They are each assigned the number 2, 3 or 4 as appropriate.

(iii) We forbid internal nodes from having incident edges in any way other than the following:

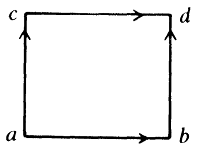


Similar prohibitions are made for the other eight categories of nodes.

(iv) We ensure a grid structure by insisting that all appropriate chains of four edges close. Thus the following scheme is forbidden:



(v) In any subgraph:



if a, b, c are internal we forbid d to be anything other than internal. Similar provisions are made for the other cases.

(vi) A bottom left corner node is compulsory.

We further ensure that the grid represents a correct computation by:

(vii) insisting that the bottom left corner represents a head position and start state, and forbidding anything else on the bottom boundary from representing a head position;

(viii) forbidding illegal transitions;

(ix) insisting on an accepting state in the top right corner.

Finally it remains to observe that the theorem holds even for a *fixed* collection B because it is sufficient to simulate just the following fixed TM, M, which is clearly complete for $\#P_1$: on unary input n, M first verifies that $n = 2pq$ (or $4pq$) where $p \leqq q$ and p and q are the ith and jth prime numbers respectively, and then simulates the ith machine in $\#\text{TIME}(n)$ on input j (or vice versa). □

The corollary above is a natural example of a problem that is provably as hard as any complete problem for the class containing it but not necessarily complete itself. Note, however, that it proves only the existence of such a problem in the sense that we cannot show A-PATTERNS to be in this category for any explicitly given A. It also remains open to determine whether Theorem 2 or the corollary holds when A or B is a singleton set.

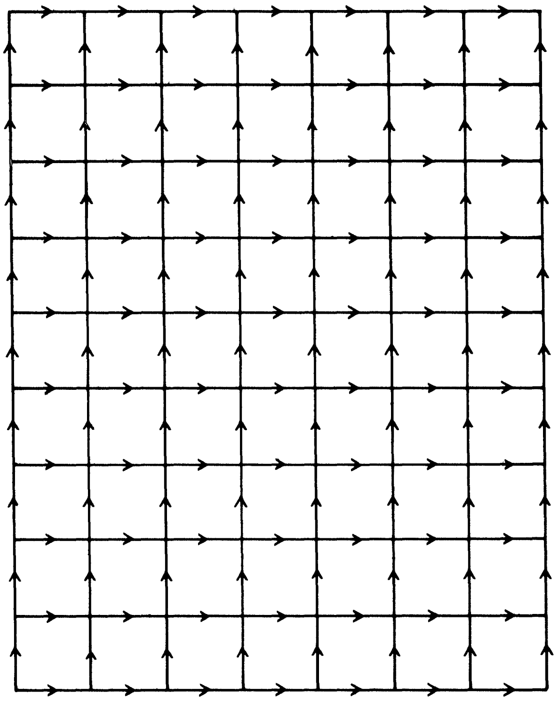


FIG. 1

6. Conclusion. We have shown that the notions of #P-completeness and of algebraic or arithmetic polynomial time reducibilities are useful tools for classifying the relative complexities of counting problems. The completeness class for #P appears to be rivalled only by that for NP in relevance to naturally occurring computational problems. Because of the richness of potential reductions it is reasonable to suppose that many further ideas will be required before the classification of new counting problems becomes a routine task.

Some possible next steps are the obvious omissions from this paper (e.g. maximal matchings, undirected trees, connectivity of all points in a graph). A more general question is that of tackling the large number of classical enumeration problems for which there is just one input for each size. For example, we have as yet no evidence that it is difficult to determine the number of Hamiltonian graphs of each size.

Acknowledgment. I am grateful to Dana Angluin for several helpful discussions. The proof given for imperfect matchings is a simplification of one found by her earlier.

REFERENCES

[1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] M. N. BARBER AND B. W. NINHAM, *Random and Restricted Walks*, Gordon and Breach, New York, 1970.

[3] M. MARCUS AND H. MINC, *Permanents*, Amer. Math. Monthly, 72 (1965), pp. 577–591.

[4] C. BRON AND J. KERBOSCH, *Algorithm* 457: *Finding all cliques in an undirected graph*, Comm. ACM, 16 (1973), pp. 575–577.

[5] S. A. COOK, *The complexity of theorem proving procedures*, Proc. 3rd ACM Symp. on Theory of Computing, 1971, pp. 151–158.

[6] D. W. DAVIES AND D. L. A. BARBER, *Communication Networks for Computers*, John Wiley, London, 1973.

[7] H. FRANK AND I. T. FRISCH, *Communication, Transmission and Transportation Networks*, Addison-Wesley, Reading, MA, 1971.

[8] F. HARARY AND E. M. PALMER, *Graphical Enumeration*, Academic Press, New York, 1973.

[9] H. C. JOHNSTON, *Cliques of a graph—Variations on the Bron–Kerbosch algorithm*, Internat. J. Comput. Information Sci., 5 (1976), pp. 209–238.

[10] R. M. KARP, *Reducibility among combinatorial problems*, Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972.

[11] P. W. KASTELEYN, *Dimer statistics and phase transitions*, J. Mathematical Phys., 4 (1963), pp. 287–293.

[12] ———, *Graph theory and crystal physics*, Graph Theory and Theoretical Physics, F. Harary, ed., Academic Press, New York, 1967.

[13] C. H. C. LITTLE, *A characterization of convertible* (0, 1)*-matrices*, J. Combinatorial Theory Ser. B, 18 (1975), pp. 187–208.

[14] M. MARCUS AND H. MINC, *On the relation between the determinant and the permanent*, Illinois J. Math., 5 (1961), pp. 376–381.

[15] E. W. MONTROLL, *Lattice Statistics*, Applied Combinatorial Mathematics, E. F. Beckenbach, ed., John Wiley, New York, 1964.

[16] T. MUIR, *On a class of permanent symmetric functions*, Proc. Roy. Soc., Edinburgh, 11 (1882), pp. 409–418.

[17] G. PÓLYA, *Aufgabe* 424, Arch. Math. Phys., 20 (1913), p. 271.

[18] H. N. V. TEMPERLEY AND M. E. FISHER, *Dimer problems in statistical mechanics—An exact result*, Philos. Mag., 6 (1961), pp. 1061–1063.

[19] L. G. VALIANT, *A polynomial reduction from satisfiability to Hamiltonian circuits that preserves the number of solutions*, Manuscript, University of Leeds, 1974.

[20] ———, *The complexity of computing the permanent*, CSR-14-77 Univ. of Edinburgh, 1977; Theor. Comput. Sci., to appear.

[21] D. ANGLUIN, *On counting problems and the polynomial time hierarchy*, Theoret. Comput. Sci., to appear.

[22] C. BERGE, *Graphs and Hypergraphs*, North-Holland, Amsterdam, 1973.

[23] R. E. TARJAN, *Enumeration of the elementary circuits of a directed graph*, this Journal, 2 (1973), pp. 211–216.