Journal of Statistical Physics, Vol. 48, Nos. 1/2, 1987

Two-Dimensional Monomer–Dimer Systems are Computationally Intractable

Mark Jerrum[1]

Received March 3, 1986; revision received February 4, 1987

The classic problem of counting monomer–dimer arrangements on a two-dimensional lattice is analyzed using techniques from theoretical computer science. Under a certain assumption, made precise in the text, it can be shown that the general problem is computationally intractable. This negative result contrasts with the special case of a system with monomer density zero, for which efficient solutions have been known for some time. A second, much easier result, obtained under the same assumption, is that the partition function of a three-dimensional Ising system is computationally intractable. Again, the negative result contrasts with known efficient techniques for evaluating the partition function of a two-dimensional system.

KEY WORDS: Computational complexity; Ising model; monomer–dimer system; $\# P$-completeness.

1. COMPUTATIONAL COMPLEXITY

The main result of this paper, in the language of the theoretical computer scientist, is that "counting matchings in a planar graph is $\# P$-complete." For readers unfamiliar with the branch of theoretical computer science known as *computational complexity*, some expansion of this stark claim is necessary. We begin, therefore, with an informal introduction to computational complexity, with the aim of enabling the nonspecialist reader to appreciate the main result, and see in what sense it justifies the claim made in the title. Precise definitions of terms will appear later, in Section 3.

One of the main goals of computational complexity is to classify problems according to their computational difficulty, and explain the apparent gulf that separates "tractable" and "intractable" problems. To

[1] Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, Scotland.

take an example from statistical physics: there is a computationally feasible method for counting dimer coverings of a large *planar* lattice, whereas all known methods for counting dimer coverings of *nonplanar* lattices rapidly break down as the lattice size grows. This apparent dichotomy is repeated in examples drawn from other fields.

Let us regard a computational problem as simply a function mapping instances to solutions (lattices to the number of dimer coverings they possess, for example). We say that a problem is *polynomial-time computable* if there is *some* algorithm (procedure) that computes this function in a length of time bounded by a polynomial in the size of the problem instance. (The size of a problem instance is the number of bits required to encode it.) In order to make this loose definition mathematically precise, it is necessary to define a formal *model of computation.* The one chosen in practice, on grounds of simplicity, is the *Turing machine.* It is important to realize that the notion of polynomial-time computability is robust under changes of machine model—a problem that is polynomial-time computable on a Turing machine is polynomial-time computable on *any* reasonable model of computation (in particular, a conventional computer), and *vice versa.* It is through polynomial-time computability that the computer scientist formalizes the notion of "tractability" referred to earlier. Counting dimer coverings of a planar lattice is an example of a polynomial-time computable problem.

It is not possible, at present, to *prove* that apparently hard problems, such as counting dimer coverings of an arbitrary lattice, are not polynomial-time computable. However, comparing the *relative* difficulty of problems can provide strong evidence for intractability. Suppose that A and B are problems to be compared. If it is possible, with the aid of a subroutine for problem B, to solve A in polynomial time (i.e., the amount of work done outside the subroutine calls is polynomially bounded), then we say that problem A is *polynomial-time (Turing) reducible* to problem B. Note that if A is reducible to B in this sense, then problem A is tractable if problem B is. The notion of reducibility should be familiar: Kirchoff, for example, *reduces* the problem of counting spanning trees in a graph to that of evaluating a certain determinant. Since determinant evaluation is tractable, so is the spanning tree problem. Many other examples could be given. For our purposes, however, it is the contrapositive of the above proposition that is pertinent: if A is polynomial-time reducible to B, then B is intractable if A is.

Before putting the idea of polynomial-time reducibility to work on counting problems, one further concept must be introduced. The class $\#P$ can be defined informally as the class of counting problems in which the structures being enumerated are easily recognized (i.e., there is a procedure, running in time polynomial in the size of the problem instance, that tests whether a putative structure has the correct form). Counting spanning trees in a graph G is an example of a problem in $\# P$; it is there because determining whether a given subgraph of G is a spanning tree of G can be done in time polynomial in the size of G. Since it is usually easy to *recognize* the things we are trying to count (even if we cannot actually count them), it is reasonable to claim that almost all the commonly studied counting problems are in the class $\# P$.

It is a remarkable fact about the class $\# P$ that it contains problems that are, in a precise sense, "as hard as any in the class." These provably hardest problems are called *$\# P$-complete.* Now, as well as containing tractable problems (such as counting spanning trees), the class $\# P$ also contains problems that are presumably intractable (such as counting dimer coverings). Thus, proving that a problem is $\# P$-complete provides substantial empirical evidence that it is computationally intractable. The concept of reducibility makes it is easy to formalize the phrase "provably hardest": A problem A in $\# P$ is $\# P$-complete iff every problem B in $\# P$ is polynomial-time reducible to A.

In practice, to prove that a problem A is $\# P$-complete, we demonstrate that some known $\# P$-complete problem is polynomial-time reducible to it. Then, since the relation "is polynomial-time reducible to" is transitive, it follows that *every* problem is $\# P$ reducible to A, and hence A is $\# P$-complete.

Those seeking a precise and formal development of these ideas are referred to Garey and Johnson.[4]

2. MONOMER–DIMER SYSTEMS

Monomer–dimer systems have been proposed by several authors[3,5] as an idealized model for certain physical systems involving diatomic molecules. A *monomer–dimer system* is defined by a space lattice L containing N sites and a number $N_d \leqslant \frac{1}{2}N$ of *dimers.* A *monomer–dimer arrangement* is an assignment of the N_d dimers to adjacent pairs of lattice sites such that no two dimers occupy a common site. The $N-2N_d$ sites left uncovered are the *monomers* of the system, and the ratio $(N-2N_d)/N$ is referred to as the *monomer density.* A *dimer system* is the special case of a monomer–dimer system in which the monomer density is 0.

The thermodynamic properties of a monomer–dimer system depend crucially on the number of possible monomer–dimer arrangements. Because of this, efficient methods of computing the number of arrangements for a given lattice L and given monomer density are of substantial physical interest. In 1961, working independently, Fisher,[1] Kasteleyn,[8] and Temperley and Fisher[12] provided an elegant and computationally feasible solution for systems based on a two-dimensional lattice and zero monomer density. (Two-dimensional systems correspond physically to the adsorbtion of diatomic molecules onto the surface of a crystal.) These authors demonstrated that the number of dimer arrangements can be written as the determinant of an $N \times N$ matrix whose entries depend in a simple way on the lattice L.[9,10] The resulting determinant can readily be evaluated by any of the usual efficient techniques, for example, Gaussian elimination. For convenience, we shall refer to this reduction as the FKT method.

We may rephrase the dimer problem in graph-theoretic terms by abstracting from the lattice L an undirected graph G whose vertices correspond to the sites of L and whose edges correspond to the pairs of adjacent lattice sites. Observe that dimer coverings of the lattice L correspond precisely to 1-factors in the graph G. (Refer to Section 3 for graph-theoretic terminology.) The FKT method solves the following graphical enumeration problem: Given an undirected planar graph G, what is the number of 1-factors of G? In applications to dimer systems, the graph G will have a regular, periodic structure. It is important to note, however, that the FKT method does not exploit periodicity, and is valid for *arbitrary* planar graphs.

Ideally, we should like to extend the FKT method in two directions:

1. By allowing the graph G to be nonplanar (corresponding physically to lattices that are three-dimensional).
2. By counting matchings in G of specified cardinality (corresponding physically to counting monomer–dimer arrangements with specified, nonzero monomer density).

Despite years of study, neither of the relaxations has found an efficient computational solution. The apparent computational difficulty of extension 1 was in large measure explained by Valiant,[13,14] who showed that the problem of computing the number of 1-factors in an arbitrary graph is $\#P$-complete.

The aim of this paper is to provide evidence that extension 2 is also computationally intractable. More precisely, we shall prove that the problem of computing the number of matchings in a planar graph is $\#P$-complete—it follows from this that computing the number of matchings of *given size* in a planar graph is also $\#P$-complete. Thus, it is unlikely that any general, computationally feasible method exists for counting monomer–dimer arrangements with given nonzero monomer density, even when the underlying lattice is two-dimensional.

Although the statement that a problem is $\#P$-complete is mathematically precise, some care is needed in its interpretation. Section 6 discusses the ways in which the evidence we present for the intractability of the monomer–dimer problem is less strong than we should ideally like.

The main result of this paper has previously appeared, in a somewhat different form, and with a more labored proof, in the author's Ph. D. thesis.[7]

3. NOTATION AND TERMINOLOGY

Let $G = \langle V, E \rangle$ be an undirected graph. A subgraph of G is *spanning* if it includes all the vertices of G. A *1-factor* of G is a spanning subgraph of G in which each vertex has degree 1. A *matching* of G is a spanning subgraph of G in which each vertex has degree 0 or 1. If M is a matching of G, then the vertices of G that have degree 1 in M are said to be *covered* by M. The following counting problems are of particular interest in the context of monomer–dimer systems:

1-FACTORS
Instance: Undirected graph G.
Output: The number of 1-factors of G.

PLANAR MATCHINGS
Instance: Undirected planar graph G.
Output: The number of matchings of G.

It will prove technically convenient to introduce a weighted version of the PLANAR MATCHINGS problem. Let $X = \{x_1, ..., x_k\}$ be a set of indeterminates, and let $\lambda: V \to X \cup \mathbb{Z}$ be a labeling function which assigns to each vertex of G either an indeterminate from X or an integer. For each matching M of G, let $w(M, G)$ denote the monomial $\prod_v \lambda(v)$, where the product is over all vertices $v \in V$ that are *not* covered by M. (The empty product is to be interpreted as 1.) The weighted version of the matching counting problem is the following:

(PLANAR) WEIGHTED MATCHINGS
Instance: A (planar) graph G with vertex labels in the set $\{-1, 0, +1\}$.
Output: $\sum_M w(M, G)$, where the sum is over all matchings M of G.

A full appreciation of the principal result of the paper requires a familiarity with the notions of *Turing machine* and *polynomial-time Turing reducibility*. The reader may refer to any text on computational complexity (e.g. Refs. 4 and 6) for an explanation of these terms. The complexity class $\#P$ is perhaps less familiar, and we shall define it here. Let Σ be a finite alphabet in which all problem instances are encoded. The class $\# P$ is a set of functions, from Σ^* to the natural numbers, specified as follows: A function $f: \Sigma^* \rightarrow \mathbb{N}$ is in $\# P$ iff there exists a polynomial-time bounded nondeterministic Turing machine T such that for all $x \in \Sigma^*$ the number of accepting computations of T, when run with x as input, is precisely $f(x)$. A counting problem (function from Σ^* to $\mathbb{N}$) is *$\# P$-complete* iff it is complete for $\# P$ with respect to polynomial-time Turing reducibility.

The class $\# P$ plays the same rôle in the analysis of counting problems as the more familiar class NP does in the analysis of decision or existence problems. As evidence of computational intractability, a demonstration that a counting problem is $\# P$-complete has the same status and flavor as a demonstration that a decision problem is NP-complete.

4. THE COMPLETENESS RESULT

Our aim is to prove that Planar Matchings is $\# P$-complete. We proceed by exhibiting a series of reductions from the problem 1-Factors, which was shown to be $\# P$-complete by Valiant.[14] Denoting the relation "is polynomial-time Turing reducible to" by the symbol $\leqslant_{\mathrm{P}}^{\mathrm{T}}$, the reductions we employ are the following:

A. 1-Factors $\leqslant_{\mathrm{P}}^{\mathrm{T}}$ Weighted Matchings.

B. Weighted Matchings $\leqslant_{\mathrm{P}}^{\mathrm{T}}$ Planar Weighted Matchings.

C. Planar Weighted Matchings $\leqslant_{\mathrm{P}}^{\mathrm{T}}$ Planar Matchings.

Since Planar Matchings is clearly in $\# P$, it will follow, by transitivity of $\leqslant_{\mathrm{P}}^{\mathrm{T}}$, that Planar Matchings is $\# P$-complete. Informally, what is shown by the reductions is that the problem Planar Matchings is computationally "at least as difficult as" the problem 1-Factors.

The first reduction is straightforward enough.

Reduction A. 1-Factors $\leqslant_{\mathrm{P}}^{\mathrm{T}}$ Weighted Matchings. Let G be an undirected graph, regarded as an instance of 1-Factors. Let $G^{(l)}$ be the labeled graph derived from G by assigning label 0 to each vertex of G. Consider the sum $\sum_M w(M, G^{(l)})$, where M ranges over all matchings of $G^{(l)}$. The only nonzero terms of the sum are those corresponding to matchings M that cover all vertices of $G^{(l)}$; moreover, all nonzero terms are in fact 1. Thus, the number of 1-factors of G is equal to $\sum_M w(M, G^{(l)})$, which is the expected output of Weighted Matchings for the problem instance $G^{(l)}$. ▌

The major work is involved in the second reduction.

Reduction B. Weighted Matchings $\leqslant_{\mathrm{P}}^{\mathrm{T}}$ Planar Weighted Matchings. Let G be an instance of Weighted Matchings, i.e., an undirected graph with vertex weights in the set $\{-1, 0, +1\}$. In outline, the reduction proceeds as follows. Embed G in the plane, identifying vertices of G with points in the plane, and edges of G with curves connecting appropriate pairs of vertices. Clearly, this can be done in such a way that all crossings of pairs of curves occur at distinct points in the plane. At each point where two edges cross, erase a small neighborhood of the intersection and insert a copy of a certain planar graph Γ; call the resulting planar graph G'. The substituting graph Γ is carefully chosen so that the weighted sum of matchings of the graph G' bears a very simple relationship to the weighted sum of matchings of G.

In presenting the construction in greater detail, it is convenient to introduce the idea of "matching polynomial" of a graph. Let $X = \{x_1,..., x_k\}$ be a set of indeterminates, and let H be an undirected graph with vertex labels drawn from the set $X \cup \mathbb{Z}$. The *matching polynomial* of H, which we shall denote by $P(H; x_1,..., x_k)$, is defined to be

$$P(H; x_1,..., x_k) = \sum_M w(M, H) \tag{1}$$

where the sum is over all matchings M of H.

The crossover graph Γ is built from the two smaller components Δ_1 and Δ_2, which are illustrated in Figs. 1 and 2. First consider the graph Δ_1. A simple computation confirms that

$$P(\Delta_1; x, y, z) = 1 + xy + yz + xz \tag{2}$$

Let us agree to refer to the degree-1 vertices of Δ_1 as *external* and the other vertices as *internal*; further, let us say that an *edge* is internal if both its endpoints are internal, and external otherwise. Equation (2) may be interpreted by considering Δ_1 in the context of some larger graph H. Suppose

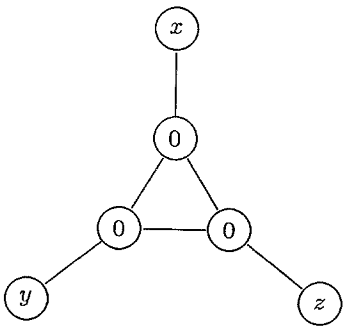

Fig. 1. The graph Δ_1.

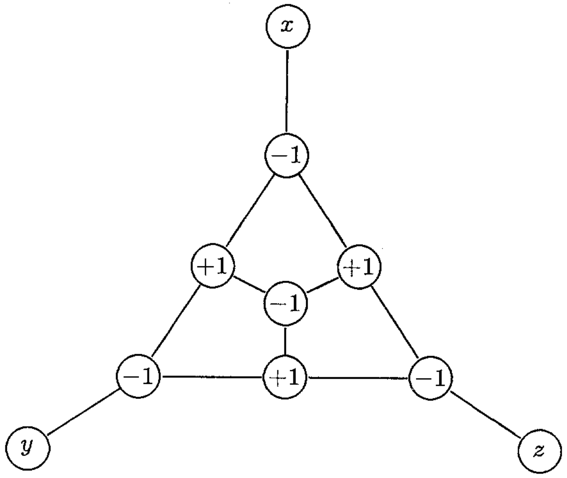

Fig. 2. The graph Δ_2.

Δ_1 is a subgraph of H that has the property that the only edges of H that are incident at internal vertices of Δ_1 are those of Δ_1 itself. Then, when computing the matching polynomial of H, we may restrict the summation in Eq. (1) to matchings M that include exactly one or three of the external edges of Δ_1; the net contribution from the remaining terms is zero.

A similar analysis may be applied to the graph Δ_2. As before, the degree-1 vertices are external, and the others internal. The matching polynomial is now

$$P(\Delta_2; x, y, z) = 2(1 + xyz) \tag{3}$$

so that when Δ_2 is considered in some larger context H, the matchings that "contribute" to the sum (1) are those that either include or exclude all the external edges of Δ_2.

The crossover graph Γ itself is illustrated in Fig. 3. Again, the degree-1 vertices are external, and the others internal. Although Γ may at first sight look complex, a second glance reveals that it is actually constructed from three copies each of Δ_1 and Δ_2. Paraphrasing our previous discussion, each copy of Δ_1 can be thought of as forcing exactly one or three of its external edges to be present in any "contributing" matching of Γ, and each copy of Δ_2 as forcing all or none of its external edges to be present. Using this computational shortcut, one can easily evaluate the matching polynomial of Γ without recourse to machine computation. In fact, we have

$$P(\Gamma; w, x, y, z) = 8(1 + xy + wz + wxyz) \tag{4}$$

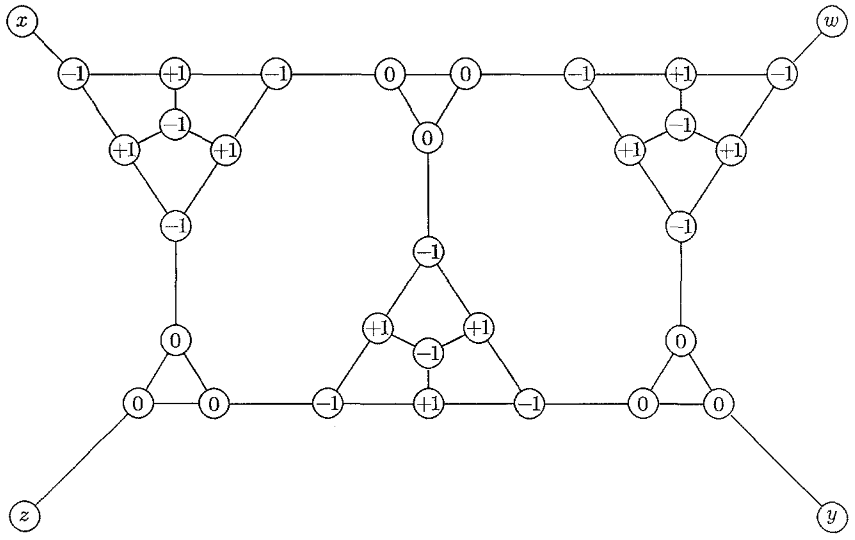

Fig. 3. The graph Γ.

This equation indicates that Γ has exactly the properties required of a crossover substitute. Placing Γ in some context H, the matchings that contribute to the sum (1) are those that do one of the following:

a. Exclude all external edges of Γ.

b. Include the external edges of Γ incident at the x- and y-labeled vertices and exclude the other two.

c. Include the external edges of Γ incident at the w- and z-labeled vertices and exclude the other two.

d. Include all external edges of Γ.

Informally, diametrically opposing pairs of external edges of Γ are forced to act in unison. (As an aside, the inspiration for the graph Γ comes from a well-known construction used in transforming arbitrary Boolean circuits to equivalent planar ones[11]; the component Δ_1 corresponds to an exclusive-or gate in the Boolean circuit construction, and the component Δ_2 to an explicit fan-out gate.)

The reduction from Weighted Matchings to Planar Weighted Matchings can now be specified. Take the undirected graph G and embed it in the plane, as suggested at the beginning of the proof. This can be done using $O(n^4)$ crossovers, where n is the number of vertices in G. Let $e_1 = \{u_1, v_1\}$ and $e_2 = \{u_2, v_2\}$ be any two edges of G that cross in the embedding. Form the graph G_1 from G by the following sequence of operations:

1. Form the (disjoint) union of G and Γ.
2. Delete the edges e_1 and e_2.
3. Identify the vertices u_1, v_1, u_2, v_2 of G with (respectively) the x-, y-, w-, and z-labeled vertices of Γ. The identified vertices acquire their labels from the graph G.

From Eq. (4) we have

$$P(G_1; \varnothing) = 8P(G; \varnothing)$$

Note that G_1 may be embedded in the plane using one fewer crossover than before. By iterating this procedure $t = O(n^4)$ times, a sequence of graphs G_1, $G_2, ..., G_t$ is produced, the final one planar and satisfying the identity

$$P(G_t; \varnothing) = 8^t P(G; \varnothing)$$

The WEIGHTED MATCHINGS problem asks us to compute the matching polynomial of G, which is easily done if the matching polynomial of the *planar* graph G_t is known. It is clear that the whole reduction is polynomial-time computable. ▮

The final reduction employs *polynomial interpolation*, a technique which appears to have central importance in establishing reductions between counting problems.[14]

Reduction C. PLANAR WEIGHTED MATCHINGS $\leqslant_P^T$ PLANAR MATCHINGS. Let $G = \langle V, E \rangle$ be an instance of PLANAR WEIGHTED MATCHINGS, i.e., an undirected, planar graph with vertex weights in the set $\{-1, 0, +1\}$. We demonstrate that the -1- and 0-labeled vertices may be eradicated at the expense of introducing extra $+1$-labeled vertices. The PLANAR WEIGHTED MATCHINGS problem for graphs in which all vertices have weight 1 is precisely the PLANAR MATCHINGS problem.

We first show how the 0-labeled vertices may be removed—a second application of the method will remove the -1-weighted vertices. Let $G^{(x)}$ be the graph derived from G by relabeling all 0-labeled vertices by the single indeterminate x. Then $P(G; \varnothing)$ is equal to $P(G^{(x)}; x)$ exaluated at $x = 0$.

Let n be the number of vertices in G. For $a = 1, 2, ..., n + 1$, define G_a to be the graph derived from $G^{(x)}$ by the following procedure:

1. For each x-labeled vertex v of $G^{(x)}$, introduce $a-1$ new 1-labeled vertices $v^{(1)},\dots, v^{(a-1)}$, and $a-1$ new edges $\{v, v^{(1)}\},\dots, \{v, v^{(a-1)}\}$.
2. Relabel each x-labeled vertex with label 1.

Let a be in the range $1 \leqslant a \leqslant n+1$, and consider any matching M of $G^{(x)}$ that leaves uncovered exactly k x-labeled vertices. The matching M may be extended in precisely a^k ways to a matching of G_a. Thus, $P(G_a; \varnothing)$ is equal to $P(G^{(x)}; x)$ evaluated at the point $x=a$. Since $P(G^{(x)}; x)$ is of degree at most n, the value of $P(G^{(x)}; x)$ at the point $x=0$ may be recovered by interpolation from the values at the points $x=1, 2,\dots, n+1$; as we have observed, these values are just $P(G_1; \varnothing),\dots, P(G_{n+1}; \varnothing)$. Using Newton's formula, one can performe the interpolation using purely integer arithmetic.

Thus, we have a reduction from a single instance of PLANAR WEIGHTED MATCHINGS to $n+1$ instances of the same problem with restricted vertex weights $\{-1, +1\}$. It is clear that a second application of the same method will eradicate the -1-labeled vertices, and hence complete the reduction to PLANAR MATCHINGS. It is also clear that the whole transformation is polynomial-time computable. ▌

5. THE ISING MODEL

The method described above in the context of monomer–dimer systems can be applied to other problems from statistical physics. It can be used, for example, to provide evidence that the partition function for three-dimensional Ising systems is computationally intractable. In contrast, a efficient procedure, due to Fisher,[2] is known for evaluating the partition function in the two-dimensional case.

Let G be an undirected graph (lattice) with N vertices (sites) and M edges (bonds). A subgraph of G is *closed* iff all its vertices have even degree (possibly zero). Denote by $C_G(z)$ the generating function for closed subgraphs of G:

$$C_G(z) = \sum_{k=0}^{\infty} c_k z^k$$

where c_k is the number of closed subgraphs of G with k edges. The *partition function* for the Ising model[9] is defined to be

$$Z(K) = (\cosh K)^M \, 2^N C_G(\tanh K)$$

We show that the problem of evaluating the coefficient of the highest degree term in the polynomial $C_G(z)$ is $\#P$-complete. It follows that the partition function itself is almost certainly hard to compute, otherwise the highest degree coefficient could be recovered by interpolation from the value of $C_G(z)$ at a small number of points.

Note that the coefficient of highest degree in C_G is simply the number of maximum closed subgraphs of G (maximum in terms of number of edges). We therefore introduce the following counting problem:

MAXIMUM CLOSED SUBGRAPHS
Instance: An undirected graph G.
Output: The number of maximum closed subgraphs of G.

The evidence for intractability of the Ising problem is captured in the following theorem:

Theorem. MAXIMUM CLOSED SUBGRAPHS is $\#P$-complete.

Proof. We first observe a correspondence between 1-factors and maximum closed subgraphs. Let $H = \langle U, A \rangle$ be an undirected graph, and suppose that each vertex of H has odd degree, and that H has at least one 1-factor. Clearly, the subgraph $\langle U, F \rangle$ is a 1-factor of H iff the complement $\langle U, A - F \rangle$ is a maximum closed subgraph of H. Thus, the number of maximum closed subgraphs of H is equal to the number of 1-factors of H.

We use the above observation to establish a polynomial-time reduction from the known $\#P$-complete problem 1-FACTORS to MAXIMUM CLOSED SUBGRAPHS. Let the undirected graph $G = \langle V, E \rangle$ be an instance of the 1-FACTORS problem. Assume that G has at least one 1-factor. (This can be tested in polynomial time by the algorithm of Edmonds.) The number of vertices of *odd* degree in any finite graph is even. Since G possesses a 1-factor, the total number of vertices in G, and hence the number of vertices of *even* degree, must be even. Denote the set of even degree verices of G by V_e. We show how to augment G in such a way that the degree of each vertex in V_e is incremented by 1.

In an arbitrary fashion, partition the vertices V_e into $\frac{1}{2}|V_e|$ unordered pairs. Take any pair of vertices $\{u, v\}$ in the partition. Form the disjoint union of G with the "3-star" graph of Fig. 4, and identify vertex u with vertex s, and v with t. Repeat this procedure with each pair of even-degree vertices in the partition, to form the augmented graph G'. Observe that

(i) Every vertex in G' has odd degree.

(ii) The number of 1-factors of G' is exactly equal to the number of 1-factors of G.

Since the 1-factors in G' are in 1–1 correspondence with the maximum closed subgraphs of G', the reduction is complete. Membership of MAXIMUM CLOSED SUBGRAPHS in $\#P$ is straightforward. ▌

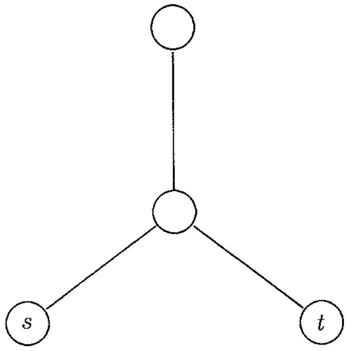

Fig. 4. The 3-star.

With a little more work, it is possible to show that evaluating the generating function $C(z)$ is hard for any *fixed* rational number z other than 1 or -1. [Evaluation of $C(-1)$ and $C(1)$ is straightforward once we observe that the closed subgraphs of G form a vector space over $GF(2)$.]

6. INTERPRETATION OF THE RESULTS

We have demonstrated that the problem of counting matchings in a planar graph is equivalent in computational difficulty to that of counting the number of accepting computations of a polynomial-time bounded nondeterministic Turing machine. Thus it is unlikely that any algorithm for PLANAR MATCHINGS exists whose run-time is bounded by a polynomial function of the size (number of vertices) of the problem instance. (Indeed, the discovery of such an algorithm would send something of a shock wave through the computer science community.) A weaker consequence of the result is that no solution in "closed form" is likely to exist; such a solution would imply the existence of a polynomial-time algorithm, but not *vice versa*.

An anonymous referee advances the following intriguing suggestion: if (as seems quite possible) the "computational power" of a physical system does not exceed that of a Turing machine, then negative results of the type presented in this paper might translate to statements about the rate of evolution of the system itself. Since the physical system could not "compute" its own evolutionary path faster than a Turing machine, one could argue that the length of time required for the system to reach its ground state must grow faster than any polynomial (in the size of the system).

The $\#P$-completeness result does, however, leave open two potential loopholes. First, the result applies to unrestricted planar graphs. It may well be that certain graphs of practical interest, particularly those with a regular structure, are amenable to solution by efficient but specialized techniques. Second, the result applies to *exact* counting of matchings. The possibility of a polynomial-time algorithm that approximates the number of matchings with relative error n^{-c} say, where c is a positive constant and n the size of the problem instance, is by no means ruled out. What can be asserted, with a reasonable degree of confidence, is that PLANAR MATCHINGS is not soluble by some analogue of the Fisher–Kasteleyn–Temperley technique for counting 1-factors.

ACKNOWLEDGMENT

I should like to thank Leslie Valiant for bringing this problem to my attention.

REFERENCES

1. M. E. Fisher, Statistical mechanics of dimers on a plane lattice, *Phys. Rev.* **124**:1664–1672 (1961).
2. M. E. Fisher, On the dimer solution of planar Ising models, *J. Math. Phys.* **7**:1776–1781 (1966).
3. R. H. Fowler and G. S. Rushbrooke, Statistical theory of perfect solutions, *Trans. Faraday Soc.* **33**:1272–1294 (1937).
4. M. R. Garey and D. S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness* (Freeman, 1979).
5. O. J. Heilmann and E. H. Lieb, Theory of monomer–dimer systems, *Commun. Math. Phys.* **25**:190–232 (1972).
6. J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, 1979).
7. M. R. Jerrum, The complexity of evaluating multivariate polynomials, Ph. D. Thesis CST-11-81, Department of Computer Science, University of Edinburgh, (1981).
8. P. W. Kasteleyn, Dimer statistics and phase transitions, *J. Math. Phys.* **4**:287–293 (1963).
9. P. W. Kasteleyn, Graph theory and crystal physics, in *Graph Theory and Theoretical Physics*, F. Harary, ed. (Academic Press, 1967), pp. 43–110.
10. J. K. Percus, *Combinatorial Methods* (Applied Mathematical Sciences 4, Springer-Verlag, 1971).
11. J. E. Savage, *The Complexity of Computing* (Wiley, 1976).
12. H. N. V. Temperley and M. E. Fisher, Dimer problem in statistical mechanics—An exact result, *Phil. Mag.* **6**:1061–1063 (1961).
13. L. G. Valiant, The complexity of computing the permanent, *Theor. Computer Sci.* **8**:189–201 (1979).
14. L. G. Valiant, The complexity of enumeration and reliability problems, *SIAM J. Computing* **8**: 410–421 (1979).