New Algorithm for the Ising Problem: Partition Function for Finite Lattice Graphs

Anna Galluccio*
Instituto di Analisi dei Sistemi ed Informatica–CNR, viale Manzoni 30, 00185 Roma, Italy

Martin Loebl[†] and Jan Vondrák[‡]
Department of Applied Mathematics, Charles University, Malostranské nám. 25, 11800 Praha, Czech Republic

(Received 12 November 1999)

We present a new efficient method to find the Ising problem partition function for finite lattice graphs embeddable on an arbitrary orientable surface, with integral coupling constants bounded in the absolute value by a polynomial of the size of the lattice graph. The algorithm has been implemented for toroidal lattices using modular arithmetic and the generalized nested dissection method. The implementation has substantially better performance than any other as far as we know.

PACS numbers: 05.50.+q, 02.10.Eb, 05.10.–a, 75.10.Hk

Max cut and the Ising model.—There are many open questions about the effects of disorder and frustration in spin glasses. The simplest spin glass system used to study these questions numerically is the Edwards-Anderson model. Its Ising version may be described as follows: A graph (or a lattice) is a pair $G = (V, E)$ where V is a finite set of *vertices* and E is a set of unordered pairs $\{u, v\} \subset V$ (*edges*). A *coupling constant* J_{ij} is assigned to each edge $\{i, j\}$; this factor characterizes the interaction between the particles represented by i and j. A physical state of the system is an assignment of spin $\sigma_i \in \{+1, -1\}$ to each vertex i. The *Hamiltonian* (or energy function) is then defined as $H(\sigma) = -\sum_{\{i,j\}\in E} J_{ij}\sigma_i\sigma_j$. The distribution of physical states over all possible energy levels is encapsulated in the *partition function* $Z(\beta) = \sum_\sigma e^{-\beta H(\sigma)}$ from which all fundamental physical quantities may be derived.

The partition function is very close to the *generating function of cuts* which is a standard concept in graph theory. For a subset of edges $A \subset E$, $w(A)$ means the sum of the weights $w(e)$ associated with the edges in A. A *cut* of a graph $G = (V, E)$ is a partition of its vertices into two disjoint subsets $V_1, V_2 \subset V$, and the implied set of edges between the two parts:

$$C(V_1, V_2) = \{\{u, v\} \in E\colon u \in V_1, v \in V_2\}.$$

The *generating function of cuts* $\mathcal{C}(G, x)$ equals the sum of $x^{w(C)}$ over all cuts C of G.

If we set the coupling constant J_{ij} as the weight $w(\{i, j\})$ of the edge $\{i, j\}$, the generating function of cuts becomes very similar to the partition function:

$$Z(\beta) = 2\sum_{\text{cut}C} e^{-\beta[2w(C)-W]} = 2e^{\beta W}\mathcal{C}(G, e^{-2\beta}),$$

where W is the sum of all the edge weights. Note that if the edge weights are integers, the generating function of cuts is a polynomial in x, while the partition function is a polynomial in e^β (both may have negative exponents).

Max cut and *min cut* are the following combinatorial optimization problems: Given a graph, divide its vertices into two parts so that the number of edges between them is as large as possible (or as small as possible). More generally, the edges may have arbitrary weights, and we want to minimize or maximize the sum of weights over all the edges between the two sets of vertices.

As the division of vertices into two parts corresponds to an assignment of positive and negative spins, and the edges in the cut are exactly those between two vertices of opposite spins, the max cut and min cut values correspond to those of maximum and minimum possible energy.

The max cut problem has its own history, but what makes it so widely studied is the enormous number of applications it finds in different fields. The general max cut problem has defied efficient solution so far, and indeed, it was proved to be *NP hard* [1], even in the case when all the edge weights are equal to ± 1. In spite of that, many attempts have been made to tackle the problem with approximation and randomized algorithms [2,3].

Polynomial time methods for toroidal square lattices were suggested in the early 1960's by Kasteleyn [4,5], and Kac and Ward [6]. Kac and Ward tried to calculate the partition function as a determinant of a $4n \times 4n$ matrix over complex numbers and even though their original derivation was not quite exact, it showed that such an approach was indeed possible. A similar method was used by Kardar and Saul [7] to obtain the partition function of a toroidal spin glass in estimated time $O(n^{3+\epsilon})$, $\epsilon < 1$. They asked whether an efficient method exists for general toroidal graphs. (This paper answers the question affirmatively.) Another polynomial time algorithm to solve the max cut problem in toroidal square lattices was proposed by Barahona in [8].

There are branch-and-cut algorithms which can be applied effectively to the case of toroidal square lattices. Barahona *et al.* [9] and De Simone *et al.* [10,11] used integer programming techniques to solve large instances of the max cut problem for toroidal square lattices, with general weights or with weights ± 1 only. The method of De Simone *et al.* [11], which has been so far the most successful one to find spin glass ground states, works in estimated time $O(n^3)$ for toroidal square lattices of n vertices, for $n \leq 50$. The method is, of course, nonpolynomial for general n. The Ising partition function of lattices of higher topological genus is also studied in [12] and [13].

However, there has been no deterministic algorithm which would produce an exact result in polynomial time for any toroidal graph, for example. Because of a recent result proved by two of the authors [14,15], though, it has become possible to solve max cut in polynomial time for any graph of genus bounded by a constant, with edge weights bounded by a polynomial of the size of the graph. The method produces actually the generating function of cuts which is equivalent to the partition function. This can be of tremendous interest for anyone studying a particular Ising model.

The aim of this paper is to describe the features of the algorithm and its implementation with emphasis on the toroidal spin glass problem. The algorithm is parallel in nature and it is based on modular arithmetic calculations. It may be particularly useful for calculation of the energy and degeneracy of the ground state and the first excited state. It makes it possible, by using massive parallelization, to determine these quantities exactly for lattices of a size up to 100×100. The time complexity of our algorithm in comparison with other methods is discussed in the last section [16].

The theory behind the algorithm.—A graph G has *genus* g if it can be drawn on an orientable surface of genus g (a sphere with g "handles" attached to it) without crossing the edges. For example, planar graphs have genus 0 and toroidal graphs have genus 1. In the following, we will consider only graphs of bounded genus, especially 0 or 1.

An *Eulerian subgraph* of a graph $G = (V, E)$ is a set of edges $U \subset E$ such that each vertex is incident with an even number of edges from U. The *generating function of Eulerian subgraphs* $\mathcal{E}(G, x)$ equals the sum of $x^{w(U)}$ over all Eulerian subgraphs U of G. There is a relation between the partition function and the generating function of Eulerian subgraphs, which was discovered by van der Waerden [17]:

$$Z(\beta) = 2^n \prod_{\{i,j\} \in E} \cosh(\beta J_{ij}) \mathcal{E}(G, \tanh(\beta J_{ij}))\,.$$

In other words, the partition function can be expressed as the generating function of Eulerian subgraphs of the same graph, with modified edge weights $w'_{ij} = \tanh(\beta J_{ij})$.

A *perfect matching* of a graph $G = (V, E)$ is a set of edges $P \subset E$ such that each vertex is incident with exactly one edge from P. The *generating function of perfect matchings* $\mathcal{P}(G, x)$ equals the sum of $x^{w(P)}$ over all perfect matchings P of G.

The generating function of Eulerian subgraphs of a graph G can be transformed into the generating function of perfect matchings of a modified graph G_s. We use Fisher's construction described in [18]. It is local in the sense that it modifies each vertex only in a way dependent on its degree and it preserves the genus of the graph. Hence we need to describe an efficient way to compute the generating function of perfect matchings of a graph of bounded genus.

Let $G = (V, E)$ be a graph on $2n$ vertices, $\{w(e), e \in E\}$ the weights assigned to the edges, and D an orientation (a fixed ordering of the two vertices of each edge). Let $A(D)$ denote the antisymmetric adjacency matrix where $a_{ij} = x^{w(i,j)}$, if (i, j) is a directed edge, $a_{ij} = -x^{w(j,i)}$, if (j, i) is a directed edge, and $a_{ij} = 0$ otherwise.

The Pfaffian of this matrix is defined as

$$\mathrm{Pf}(A(D)) = \sum_P \mathrm{sgn}(P) a_{i_1 j_1} a_{i_2 j_2} \ldots a_{i_n j_n}\,,$$

where the sum is taken over all partitions of the index set $\{1, 2, \ldots, 2n\}$ into pairs $i_1 < j_1, \ldots, i_n < j_n$ and $\mathrm{sgn}(P)$ is the sign of the permutation $(i_1, j_1, i_2, j_2, \ldots, i_n, j_n)$. The Pfaffian is similar to the determinant of a matrix and it can be calculated efficiently.

It can be observed that the nonzero terms contributing to the Pfaffian are exactly those corresponding to perfect matchings of G (partitionings of the set of vertices into pairs where there is an edge between each pair). However, each of them comes with a positive or negative sign, depending on the orientation D. What we would like to find is a special orientation that produces positive signs for all perfect matchings. Then the Pfaffian would be exactly equal to the generating function of perfect matchings. Such a special orientation (called *Pfaffian*) exists indeed for planar graphs, which was proved by Kasteleyn [4].

For general graphs we cannot rely on the Pfaffian orientation, as it may not always exist. However, the situation is solved by the *Galluccio-Loebl theorem* [14] which states that a graph G of genus g has 4^g *relevant orientations* such that a suitable linear combination of the corresponding Pfaffians equals the generating function of perfect matchings of G. To calculate the Pfaffians we use a method which was developed by George [19] and later refined by Lipton, Rose, and Tarjan [20]. Their approach, though intended for Gaussian elimination, can be geared to Pfaffian elimination as well. If G is a graph of bounded genus on n vertices, Pfaffian elimination of the corresponding adjacency matrix can be performed in $O(n \log n)$ space and $O(n^{3/2})$ time.

Here it should be noted that the units of the time complexity are actually the basic operations with elements of the matrices. However, these elements are polynomials in one variable. If we wanted to perform the operations symbolically, we could hardly hope to carry them out in constant time. Also, we would have to face the problem of numerical stability, and the memory requirements for symbolic manipulation would be enormous. Instead, we decided to use *modular arithmetic*.

For simplicity, we suppose that the edge weights (and hence the exponents in the polynomial) are integers. We perform the elimination numerically, in a finite field $GF[p]$ where p is a prime number. This means all operations are performed modulo p and we can multiply and divide without loss of precision. The operations can be performed in constant time and the elements take up constant space.

On the other hand, the result of this computation is only the value of the generating function modulo some prime. In order to reconstruct the whole polynomial, we need to calculate the values at sufficiently many points, obtain the polynomial by interpolation in the finite field, and then (if necessary) repeat the computation in several finite fields and compose the partial results to get the complete polynomial.

Another advantage of this approach is that the algorithm is parallel in nature—the computation is broken up into many independent parts which can be processed in parallel. Or we can use only the coefficients obtained in one finite field as partial results; this can be useful if we only want to find the value of the ground state energy, i.e., the first nonzero coefficient in the polynomial.

As a consequence, the partition function can be computed efficiently and precisely for any graph of bounded genus, with coupling constants bounded in the absolute value by a polynomial of the size of the graph.

Algorithm overview.—Let us summarize the algorithm to calculate the partition function for a given toroidal graph $G = (V, E)$:

(1) Find prime numbers $p_1, p_2, \ldots, p_k < 2^{16}$ so that $\prod_{i=1}^{k} p_i > 2^{|V|}$. For each of them, repeat the remaining steps of the algorithm, performing all operations in $GF[p_i]$.

(2) Choose $m + 1$ distinct elements $x_0, \ldots, x_m \in GF[p_i]$, where m is the maximum possible degree of the polynomial. (Avoid $x_j = 0$ because the elements must be invertible.) For each of them, repeat the following step.

(3) Construct the matrices encoding the relevant orientations of the modified graph G_s, with hyperbolic functions $\tanh(\beta J_{ij})$ substituted for edge weights. For every occurrence of β substitute $e^{\beta} = x_j$ and calculate the four Pfaffians. From these values, calculate the value of $Z(\beta)$.

(4) Obtain the coefficients of $Z(\beta)$ (modulo p_i) by interpolation in $GF[p_i]$; we use the fact that if we have the values of our polynomial modulo some prime p, we can also extract its coefficients modulo p.

(5) Apply the Chinese remainder theorem and compose the results for each coefficient from all the finite fields, to obtain the complete partition function.

The number of finite fields is $O(n)$, assuming there is a sufficient number of primes in the range where our hardware is able to perform modular arithmetic in constant time. Because of this constraint, the algorithm is actually unable to work properly for an arbitrarily large input, and any complexity analysis in the sense of asymptotic behavior is meaningless. On the other hand, the limit on the size of lattices we are able to process is well beyond our practical possibilities. The product of all primes below 2^{16} is approximately $2^{2^{16}}$, which means we can process lattices with at most 2^{16} vertices. If this should prove insufficient, we could still move to 32-bit arithmetic and the limit of 2^{32} vertices which would be enough to keep all our computing resources busy for more than the lifetime of our Universe.

If the edge weights are bounded by a constant, the number of evaluations in each of the fields is $O(n)$. (In general, the number of values is polynomial if the edge weights are bounded by a polynomial of the size of the lattice.) A single evaluation of the polynomial takes $O(n^{3/2})$, so the computation in each finite field takes $O(n^{5/2})$ time. The total time complexity of step 3 (under the restrictions mentioned above) is therefore $O(n^{7/2})$. The interpolation in step 4 and the final composition in step 5 take $O(n^3)$ time in total, so they are faster than the Pfaffian evaluation in step 3.

Steps 2, 3, and 4 can be parallelized easily. The computation in each of the finite fields can be performed separately, and the communication is trivial: in step 5, we only have to send the results modulo each of the primes, i.e., data of size $O(n^2)$. With this degree of parallelization [$O(n)$ processors are needed], steps 2, 3, and 4 take $O(n^{5/2})$ time, whereas step 5 takes $O(n^3)$. To remove this obstacle, we could parallelize it as well; every processor would produce one of the $O(n)$ coefficients in $O(n^2)$ time. Then the total (parallel) time complexity would be $O(n^{5/2})$.

As mentioned above, the computation in each of the finite fields takes $O(n^{5/2})$ time. The product of this part of the algorithm is the partition function with coefficients modulo p_i. The information we obtain here is significantly reduced, but we can still use it for detection of the ground state energy (which corresponds to the first nonzero coefficient in the partition function).

If the remainder of a coefficient modulo a prime number is nonzero, then the coefficient is certainly nonzero. On the other hand, if the remainder is zero, the original coefficient is zero with high probability. By allowing ourselves to choose from a wide enough range of primes, we can make the chance of error arbitrarily small. We can also reduce the probability of error by performing independent computations in several finite fields.

The time complexity $O(n^{5/2})$ in one finite field compares favorably with the estimated complexity $O(n^3)$ of the most successful method to calculate the ground state energy which uses integer programming techniques [11]. As regards the complete partition function, the complexity of our algorithm is comparable with the estimated complexity of the implementation of Kardar and Saul [7]. Apart from our method, this is the only method providing the exact spin glass partition function for toroidal square lattices that we know about. However, the advantages of our algorithm are its self-contained structure and easy parallelization. Moreover, as mentioned above, computation in one or a few finite fields yields the values of some coefficients with high probability.

In order to test the software we carried out a number of experiments on square toroidal lattices with random coupling constants ± 1. For illustration, the running times (sequentially on a PC with an Athlon/500 MHz CPU) are as follows: The complete partition function for a 10×10 lattice takes 15 sec to compute, a 20×20 lattice 25 min, a 30×30 lattice 8 h, and a 50×50 lattice 14 days. The computation in one finite field (which is practically sufficient for ground state detection) takes only 2 h for the 50×50 lattice. In comparison, Kardar and Saul [7] report a computation time of 110 sec for a 10×10 lattice, on an Indigo 4000 workstation. The values for larger lattices are not provided in their paper. We also computed the regular Ising energy histograms for a 32×32 lattice (all coupling constants equal to $+1$). The result matches that of Beale [21].

We would like to thank Jirka Matoušek and Giovanni Rinaldi for helpful discussions. A. G. was supported by NATO-CNR and M. L. by FONDAP on applied mathematics, GAUK 158/99.

*Electronic address: galluccio@iasi.rm.cnr.it
†Electronic address: loebl@kam.ms.mff.cuni.cz
‡Electronic address: vondrak@kam.ms.mff.cuni.cz

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability* (W. H. Freeman and Company, New York, 1979).
[2] S. Poljak and F. Rendl, Discr. Appl. Math. **62**, 249–278 (1995).
[3] M. X. Goemans and D. P. Williamson, Assoc. Comput. Machinery **42**, 1115–1145 (1995).
[4] P. W. Kasteleyn, *Graph Theory and Theoretical Physics* (Academic Press, New York, 1967).
[5] P. W. Kasteleyn, J. Math. Phys. **4**, 287–293 (1963).
[6] M. Kac and J. C. Ward, Phys. Rev. **88**, 1332–1337 (1952).
[7] M. Kardar and L. Saul, Nucl. Phys. B **432**, 641–667 (1994).
[8] F. Barahona, J. Phys. A **15**, 3241–3253 (1982).
[9] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, Oper. Res. **36**, 493–513 (1988).
[10] C. De Simone, M. Diehl, M. Jünger, P. Mützel, G. Reinelt, and G. Rinaldi, J. Stat. Phys. **80**, 487–496 (1995).
[11] C. De Simone, M. Diehl, M. Jünger, P. Mützel, G. Reinelt, and G. Rinaldi, J. Stat. Phys. **84**, 1363–1371 (1996).
[12] T. Regge and R. Zecchina, J. Math. Phys. **37**, 2796 (1996).
[13] T. Regge and R. Zecchina, J. Phys A **33**, 741–761 (2000).
[14] A. Galluccio and M. Loebl, Electron. J. Combinatorics **6**, R6 (1999).
[15] A. Galluccio and M. Loebl, Electron. J. Combinatorics **6**, R7 (1999).
[16] The program is freely available from the authors. If you are interested, please send an email message to: vondrak@kam.ms.mff.cuni.cz
[17] B. L. van der Waerden, Z. Phys. **118**, 473 (1941).
[18] M. E. Fisher, J. Math. Phys. **7**, 10 (1966).
[19] A. George, SIAM J. Numer. Anal. **10**, 345–363 (1973).
[20] R. J. Lipton, D. J. Rose, and R. E. Tarjan, SIAM J. Numer. Anal. **16**, 346–358 (1979).
[21] P. D. Beale, Phys. Rev. Lett. **76**, 78–81 (1996).