

A classical approximation scheme for the ground-state energy of Ising spin Hamiltonians on planar graphs

Nikhil Bansal, Sergey Bravyi and Barbara M. Terhal

IBM Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598, USA

(Dated: May 8, 2007)

We show that there exists a classical PTAS (Polynomial Time Approximation Scheme) for the ground-state energy problem of classical Hamiltonians which consist of interactions between the vertices of a planar graph. The problem of determining the ground-state energy exactly is known to be NP-complete for classical Hamiltonians on a 2D square lattice. We discuss the classical and possible quantum running times of our schemes.

INTRODUCTION

The paradigm of adiabatic quantum computation was introduced in [1]. In adiabatic quantum computation one considers a quantum system at zero temperature. At time $t = 0$ the system is in the ground-state of some simple Hamiltonian H_0 . The computation proceeds by slowly varying this Hamiltonian in time, up to time $t = T$. If this variation happens at sufficiently slow speed, the adiabatic theorem (see [2] for a most accessible derivation of this theorem) guarantees that the system remains close to the ground-state of the instantaneous Hamiltonian at all times. The requirements for slowness of the variations and hence for the total running time of the computation depend crucially on the gap $\Delta(t)$ between the ground-state energy $\lambda_0(t) \equiv \lambda(t)$ and the energy of the first excited state $\lambda_1(t)$ at time t . If for a particular adiabatic path this gap $\Delta(t) \geq \frac{1}{\text{poly}(n)}$ where n is the number of qubits in the system and $\text{poly}(n)$ is some polynomial in n , then it can be shown that the total running time T of the adiabatic procedure will also be some polynomial in n . Algorithms with a running time that is polynomial in the problem size are called *efficient*, in contrast with inefficient procedures that take super-polynomial or exponential running times. Let us consider the following adiabatic path that is typically considered in the literature. For $t \in [0, T]$ we have

$$H_t = \left(1 - \frac{t}{T}\right) \sum_{u=1}^n X_u + \frac{t}{T} H(Z). \quad (1)$$

Here X_u is the Pauli X -matrix acting on qubit u tensored with the identity matrix on all other qubits. The Hamiltonian $H(Z)$ is a sum of terms each of which is proportional to $Z_{u_1} Z_{u_2} \dots Z_{u_k}$ where Z_u is the Pauli Z -matrix on qubit u tensored with I elsewhere. In other words, we can view $H(Z)$ as a *classical* Hamiltonian that consists of sums of interactions between at most k classical spins. If the energy gap $\Delta(t)$ of H_t were $\frac{1}{\text{poly}(n)}$ at all times t then such adiabatic computation would result in the ground-state of the final Hamiltonian $H_T = H(Z)$ which is a classical spin state. It is not hard to show that one can encode the answer to a NP-complete problem in

this final state and its associated energy. An example is the representation of the problem 3-SAT (for a definition of this problem see [3]) as a collection of 3-local terms of $H(Z)$ [4]. Thus a $\frac{1}{\text{poly}(n)}$ lower-bound on the gap at all times t would give rise to an efficient quantum algorithm for some NP-complete problem. Even though such perspective may have seemed tantalizing 10 years ago, there is now a growing consensus that it is unlikely that a quantum computer could pull off such feat. This consensus should be viewed as an extension to the equally unproven but largely accepted belief that classical computers cannot solve NP-complete problems efficiently. The main intuition behind this belief is our common experience of the disparity between the relatively easy task of *verifying* the correctness of something, a proof of a theorem, the solution of a riddle, the optimal solution to a NP-complete problem, versus the task of providing the proof or solution in the first place [5].

Thus if we assume the validity of the conjecture that quantum nor classical computers can solve NP-complete problems, it follows that a quantum adiabatic computation such as in Eq. (1) is a means to obtain an *approximation* to the ground-state and the ground-state energy. An example of an adiabatic computation as in Eq. (1) is one in which the Hamiltonian $H(Z)$ is the Hamiltonian of a classical Ising spin glass defined on a lattice or graph. Let $G = (V, E)$ denote any graph with n vertices and let $u, v \in V$ be vertices of the graph. We write $(u, v) \in E$ to mean that there is an edge between u and v in G . For each vertex u , we have a classical spin variable S_u which can take values ± 1 . We can define a classical Hamiltonian with interaction graph G as

$$H(S) = \sum_{(u,v) \in E} c_{uv} S_u S_v + \sum_{u \in V} d_u S_u. \quad (2)$$

Here c_{uv} and d_u can be given as some m -bit numbers. The function $H(S)$ is obtained from the operator $H(Z)$ by replacing the Pauli Z -matrix at vertex u by the variable S_u . Bieche et al. [6] has shown that for planar graphs G the problem of determining the minimum value of $H(S)$ (the ground-state energy) and the associated assignment S can be solved efficiently if there are no linear terms (i.e. all $d_u = 0$). Barahona [7] showed that in the

presence of linear terms determining the minimum value of $H(S)$ is NP-complete. Recently the company D-wave has claimed to have implemented this spin glass Hamiltonian and the adiabatic evolution of Eq. (1) for 16 qubits on a 4×4 square lattice, see [8].

In order to understand the possible power of adiabatic computation, it is thus of interest to consider how well an approximation to the ground-state energy can be obtained by purely classical algorithmic means. The area of approximation algorithms is an active area of research in computer science, see e.g. [9]. Three main types of approximations to optimization problems can be distinguished. A problem is said to have a polynomial time approximation scheme (PTAS) if given any $\epsilon > 0$, there is an algorithm A_ϵ , that for any instance I produces a solution within $(1 \pm \epsilon)$ times the optimal solution, and has a running time which is a polynomial in the input size of I . Observe that the running time of a PTAS is only required to be polynomial in input size, and it can have an arbitrary dependence on ϵ (for example $n^{1/\epsilon}$ or $2^{1/\epsilon} \text{poly}(n)$ are valid running times for a PTAS). A stronger notion is that of a fully polynomial time approximation scheme (FPTAS), where the running time of the approximation scheme is required to be polynomial both in the size of the input and in $(1/\epsilon)$. For the Ising spin glass problem an instance is a particular graph and set of values of the weights c_{uv} and d_u . The optimal solution is the minimum value of the energy $H(S)$.

For the Ising spin glass on planar graphs with coupling c_{uv}, d_u specified by a constant number of bits, we can exclude the possibility of a FPTAS (assuming that classical computers cannot solve NP-complete problems). Assume we have a FPTAS and set $\epsilon = \delta/\text{poly}(n)$ for some constant δ . This gives polynomial-time algorithm to approximate $\min_S H(S)$ with an error which is at most $\delta \min_S H(S)/\text{poly}(n)$. This is sufficient accuracy to solve the NP-complete problem *exactly* since the difference between the minimum of $H(S)$ and the value right above it (i.e. the energy gap) is a constant, that is, independent of n . Therefore any quantum adiabatic computations such as the one in Eq. (1) is unlikely to provide a FPTAS for the Ising spin glass problem.

Few NP-hard problems admit a PTAS. A problem is called APX-hard, if there exists a fixed constant $\delta > 0$, such that a $(1 + \delta)$ polynomial time approximation would imply P=NP. It is known that no PTAS can exist for the Ising problem with only quadratic terms for general graphs, see the references and results in [10]. One can even get an inapproximability result for cubic graphs, where all edge weights are +1 or -1 and all linear terms are 0 by using the reduction from Max-Cut in [11], and observing that Max-Cut is APX-hard for cubic graphs [12].

In this Letter we will show that there exists a classical PTAS for the Ising spin glass problem in Eq. (2) when the graph G is planar.

In a future paper we will show how to extend these results to quantum Hamiltonians on planar graphs [13].

We first consider the case when G is a two-dimensional lattice, and give a PTAS for this case. The result easily extends to any constant-dimensional lattice. We then describe a more general technique which implies a PTAS for arbitrary planar graphs. In all our results the weights d_u and c_{uv} are allowed to be completely arbitrary (but representable on a computer using, say, m bits).

All our algorithms work by first decomposing the graph into smaller and simpler graphs, solving the problem on these simpler graphs, and then patching the partial solutions to obtain a solution to the original problem. We note that this technique is fairly standard for solving hard problems on planar graphs. In fact, many problems admit a PTAS on planar graphs even though approximating them on general graphs is known to be NP-hard, see [14, 15, 16]. While our techniques are similar to those of [15] and [16] at a conceptual level, our results do not follow directly from their work and require some new ideas. The main difficulty is that the Hamiltonian involves both positive and negative terms which can possibly cancel out.

Note that the existence of a PTAS for a Hamiltonian $H(S)$ does not imply the existence of a PTAS for the trivially related problem of finding the minimum of $H(S)+C$; this is because the PTAS produces a solution with small *relative* error.

THE LATTICE CASE

We begin with a simple property of the optimal solution (the one that minimizes $H(S)$) that holds for an arbitrary graph. Let us write the Hamiltonian as $H(S) = Q(S) + L(S)$, where $Q(\cdot)$ is the contribution of the quadratic (2-local) terms and $L(\cdot)$ is the contribution of the linear (1-local) terms.

Claim 1. *There exists an optimal solution S_{opt} such that $H(S_{\text{opt}}) \leq 0$ and $L(S_{\text{opt}}) \leq 0$.*

Proof. It is clear that $\sum_S H(S) = 0$. This implies that there must exist a spin configuration with negative energy and thus $H(S_{\text{opt}})$ is negative. For the second part, suppose on the contrary that $L(S_{\text{opt}}) > 0$. Then consider the solution $-S_{\text{opt}}$ which is obtained by replacing each S_u by $-S_u$. This solution has an unchanged quadratic contribution and the linear term is now negative. So it cannot be that $L(S_{\text{opt}}) > 0$. \square

From now on S_{opt} will denote an optimal solution such that $L(S_{\text{opt}}) \leq 0$.

Decomposing the problem Let $\epsilon > 0$ be a fixed small constant. Without loss of generality let us assume that $t = 1/\epsilon$ is an integer. For $i = 0, \dots, t-1$, let X_i denote

the set of vertices $u = (x, y)$ on the horizontal lines defined by $\{y \equiv i \pmod{t}\}$. Similarly, let Y_i denote the set of vertices on the vertical lines defined by $\{x \equiv i \pmod{t}\}$. Let E_{ij} denote the set of edges that have at least one endpoint in X_i or Y_j . Consider Hamiltonians

$$Q_{ij}(S) = \sum_{(u,v) \in E_{ij}} c_{uv} S_u S_v, \quad L_{ij}(S) = \sum_{u \in X_i \cup Y_j} d_u S_u.$$

Define also $H_{ij}(S) = Q_{ij}(S) + L_{ij}(S)$.

Lemma 2. *There exists a choice of i and j such that $H_{ij}(S_{opt}) \geq 3\epsilon H(S_{opt})$ and thus*

$$\min_{i,j} \min_S (H(S) - H_{ij}(S)) \leq (1 - 3\epsilon)H(S_{opt}). \quad (3)$$

Proof. Consider all the sets E_{ij} for $0 \leq i, j \leq t-1$. Each edge lies in exactly $t^2 - (t-1) \times (t-2) = 3t - 2$ such sets E_{ij} . Analogously, each vertex lies in exactly $t^2 - (t-1)^2 = 2t - 1$ sets $X_i \cup Y_j$. Thus we have $\sum_{i=0}^{t-1} \sum_{j=0}^{t-1} H_{ij}(S_{opt}) = \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} Q_{ij}(S_{opt}) + L_{ij}(S_{opt}) =$

$$\begin{aligned} (3t-2)Q(S_{opt}) + (2t-1)L(S_{opt}) &\geq \\ (3t-2)(Q(S_{opt}) + L(S_{opt})) &= (3t-2)H(S_{opt}). \end{aligned}$$

The inequality follows by Claim 1 as $L(S_{opt}) \leq 0$. Thus, $\exists i, j$ such that $H_{ij}(S_{opt}) \geq (3t-2)H(S_{opt})/t^2 \geq 3\epsilon H(S_{opt})$, since $H(S_{opt}) \leq 0$. Therefore the minimum in Eq. (3) is at most $H(S_{opt}) - H_{ij}(S_{opt}) \leq (1 - 3\epsilon)H(S_{opt})$. \square

Consider the Hamiltonian $H^{ij}(S) = H(S) - H_{ij}(S)$. Let S'_{opt} be a spin assignment that achieves the minimum in Eq. (3). Note that H^{ij} splits into non-interacting blocks of spins of size $t \times t$, so S'_{opt} can be found efficiently (see below).

Lemma 3. *One can choose the optimal solution S'_{opt} such that $H_{ij}(S'_{opt}) \leq 0$ and thus $H(S'_{opt}) \leq H^{ij}(S'_{opt})$.*

Proof. The value of $H^{ij}(S)$ does not depend upon S_u at vertices $u \in X_i \cup Y_j$. We can use this freedom in choosing such S_u to make $H_{ij}(S'_{opt}) \leq 0$. Indeed, let us color the vertices of the lattice by white and black in chess order. Given any optimal solution S'_{opt} , define four total spin configurations $S^{(a)}$, $a = 1, 2, 3, 4$ by setting the spins $u \in X_i \cup Y_j$:

$S^{(1)}$: Set all $S_u = +1$.

$S^{(2)}$: Set all $S_u = -1$.

$S^{(3)}$: Set $S_u = +1$ for white u and $S_u = -1$ for black u .

$S^{(4)}$: Set $S_u = -1$ for white u and $S_u = +1$ for black u .

All configurations $S^{(a)}$ coincide with S'_{opt} whenever $u \notin X_i \cup Y_j$. Note that $\sum_{a=1}^4 Q_{ij}(S^{(a)}) = 0$ since every edge in E_{ij} has one black and one white endpoint. Besides,

$\sum_{a=1}^4 L_{ij}(S^{(a)}) = 0$ since each spin in $X_i \cup Y_j$ takes value $+1$ for exactly two assignments $S^{(a)}$. We conclude that $\sum_{a=1}^4 H_{ij}(S^{(a)}) = 0$. Therefore one can choose one of the four configurations such that $H_{ij}(S^{(a)}) \leq 0$. Since we have not changed the spins at vertices $u \notin X_i \cup Y_j$, one has $H(S^{(a)}) = H^{ij}(S^{(a)}) + H_{ij}(S^{(a)}) \leq H^{ij}(S^{(a)}) = H^{ij}(S'_{opt})$. \square

As a conclusion of Lemmas 2,3 our PTAS generates a solution S'_{opt} for all spins in the lattice such that $H(S'_{opt}) \leq (1 - 3\epsilon)H(S_{opt})$. What is the running time of our algorithm? The algorithm works as follows. It tries out each of the possible choices of i and j for which to decompose the lattice. For each such choice, it decomposes the lattice into blocks, computes the optimum block solution, and then tries one of four solutions on the boundary to check which one works. Thus the running time is $\frac{1}{\epsilon^2} \cdot T_{\text{block}}(\frac{1}{\epsilon}, \frac{1}{\epsilon}) \cdot \epsilon^2 n$ where $T_{\text{block}}(r, s)$ is the time to find the optimal solution for a subblock of size $r \times s$. For such $r \times s$ subblock one can do a brute force search through all possible assignments in time $O(2^{rs})$. However, a better method that is commonly employed in approximation algorithms is to use dynamic programming which uses more (storage) space but less time. One can easily show that

Lemma 4. *There is a dynamic programming algorithm that computes the optimum solution for a $r \times s$ lattice-block using space $O(r2^s)$ and time $O(r4^s)$.*

Proof. Let B_i denote the lattice B restricted to row i or less ($B_r = B$). Let S be $\{-1, +1\}$ vector of size s and for $i \in \{1, \dots, r\}$, let $V(i, S)$ be the optimum value of the Hamiltonian restricted to B_i such that the variables on the i -th row have assignment S . The dynamic program computes and stores $V(i, S)$ for all i and S starting sequentially from $i = 1$. This suffices as the optimum solution for B is exactly $\min_S V(r, S)$.

For $i = 1$, the quantity $V(1, S)$ can be easily computed for each S . Suppose $V(i, S)$ has been computed and stored for all S . For an assignment S of row $i + 1$ and an assignment S' of row i , let $Z(i + 1, S, S')$ denote the contribution of all terms to the Hamiltonian corresponding to vertices on row $i + 1$, and edges in B_{i+1} with at least one end point in row $i + 1$. Since the assignment S on row $i + 1$ can only affect Z , it follows that

$$V(i + 1, S) = \min_{S'} (V(i, S') + Z(i + 1, S, S')). \quad (4)$$

Since $Z(i + 1, S, S')$ can be computed in time $O(s)$ and $V(i, S')$ are already stored, computing $V(i + 1, S)$ for all S takes time $O(s2^s \cdot 2^s)$. We can speed up this procedure somewhat to $O(4^s)$ by considering the assignments S' in say the Gray code order (where successive assignments differ in exactly 1 variable), and hence only $O(1)$ work needs to be done per assignment S' . \square

Thus using the dynamic program on the blocks leads to an overall running time of the PTAS of $O(n4^{3/\delta}/\delta)$ where δ is the relative error. Let us mention that the running time can be improved to $O(n4^{1/\delta})$ if one partitions the lattice into strips rather than square blocks. For the partition into strips the analogue of the Hamiltonians $H_{ij}(S)$ are Hamiltonians $H_i^x(S)$ ($H_j^y(S)$) that include all terms $d_u S_u$, $u \in X_i$ ($u \in Y_j$) and all terms $c_{uv} S_u S_v$ such that (u, v) is a vertical (horizontal) edge that has exactly one end-point in X_i (Y_j). Note that $\sum_{i=1}^{\ell-1} H_i^x(S) + H_i^y(S) = 2H(S)$. This decomposition leads to an analogue of Lemma 2 with a relative error ϵ instead of 3ϵ .

Let us consider how a quantum computer could improve these running times. For the simple classical method in which we solve the subblock problem by searching through all possible solutions, it is clear that one can obtain a speed-up by applying the quantum minimum function algorithm [17] which is based on Grover's search algorithm. This implies a quantum running time of $O(n2^{9/2\delta^2})$ which is *worse* than the running time of the dynamic programming method on the disconnected strips. Note however that this quantum algorithm would use only little space scaling as the size of the sub-lattice (and not exponential in this size). It is possible that could implement this algorithm as an adiabatic quantum computation.

Another possible application of quantum searching is obviously in the dynamic programming. Since many classical approximation algorithms rely on dynamic programming, this could be an important area of applications. There is a catch however with this use of Grover's algorithm. We could use a quantum algorithm in order to perform the minimization on the r.h.s. of Eq. (4). The 2^s values of $V(i, S')$ for each S' are *stored* which implies that this is a problem of searching in a real database. In other words which item is 'marked' is not determined by a simple function call (alternatively, the function $V(i, S')$ that would have to be called is nonlocal in S' and depends on the whole sublattice up till row i). In the database setting one has to consider the additional time/hardware overhead in accessing the spatially extended database. Optical or classical wave implementations of this type of searching have been considered, see [18, 19]. One can use the Grover search subroutine for solving the lattice strips and one obtains a running time of $T_q = O(n2^{3\delta/2})$ not taking into account the additional overheads. Whether this application of Grover's algorithm in dynamic programming is of genuine interest will depend how its physical implementation competes in practice with the capabilities of classical computers.

Let us briefly sketch how our results generalize to d -dimensional lattices. The lattice can be decomposed into blocks such that each block has length $1/\epsilon$ along $d - 1$ dimensions. There is some choice of decomposition such that the value of the Hamiltonian restricted to these

blocks is at most $(1 - \epsilon(2d - 1)(d - 1)/d)H_{\text{opt}}$. Choosing a random assignment on the removed vertices ensures that the value of the total Hamiltonian solution is at most that of the Hamiltonian on the blocks. Solving the problem for blocks using dynamic programming requires a total time of $O(n\epsilon^{-1}4^{(1/\epsilon)^{d-1}})$.

GENERAL PLANAR GRAPHS

Let us now consider the general case of planar graphs. In the lattice case we used the symmetry of the lattice to argue that there exists a small subset of edges and vertices such that they have a small contribution to the Hamiltonian and removing them decomposes the lattice into small disjoint blocks. For general planar graphs we cannot use this argument due to the lack of symmetry and hence we argue indirectly. We show that the magnitude of the optimal solution is at least a constant fraction of the sum of the absolute values of quadratic terms corresponding to the edges. This allows us to find a subset of edges with relatively small weight such that removing them decomposes the graph into simpler disjoint graphs for which the problem can be solved directly. Since the removed edges have small weight adding them back in does not increase the Hamiltonian by too much, irrespective of how the values of the vertices incident to these edges are assigned.

For a planar graph $G = (V, E)$ let $C = \sum_{(u,v) \in E} |c_{uv}|$. The key to our PTAS is the following result that shows that the value of the optimal assignment scales linearly with C .

Theorem 5. *If G is planar, then $H(S_{\text{opt}}) \leq -C/3$.*

We will prove this Theorem in the Appendix. Given this Theorem the PTAS follows from the ideas of [16]. We sketch the details here for completeness. We begin by describing the crucial notions of p -outerplanar graphs and tree-widths.

Definition 6 (p -outerplanar graphs). *A 1-outerplanar graph is a planar graph that has an embedding in the plane with all vertices appearing on the outerface. A p -outerplanar graph is a planar graph that has an embedding in the plane such that removing all the vertices on the outer face gives a $(p - 1)$ -outerplanar graph.*

Note that the outerface of a 1-outerplanar graph need not be a simple cycle. A tree or the graph consisting of two cycles that share a common vertex are both 1-outerplanar.

The notion of tree decompositions and tree-width was introduced by Robertson and Seymour [20] and is extremely useful. We refer the reader to [16, 21] for the relevant definitions and the intuition behind them. It

is known that a p -outerplanar graph has a tree-width of at most $3p - 1$ [22]. The tree decomposition of a p -outerplanar graph can be computed in time $O(pn)$ [23]. Given the tree decomposition, the Ising problem can be solved exactly in time $O(n2^k)$ on graphs with n vertices and of tree-width k using standard techniques (see for example [16, 21]).

Our algorithm works as follows. Given the planar graph G , it first constructs a drawing of G in the plane. This can be done in linear time using for example the algorithm of Hopcroft and Tarjan [24]. This gives an outerplanar decomposition of G . Say it is h -outerplanar (h could be as large as $O(n)$). Partition the vertices into levels V_1, \dots, V_h where V_1 is the outer face and V_i is the outer face obtained by removing V_1, \dots, V_{i-1} . Let E_i be the edges that go from V_i to V_{i+1} . For $j = 0, \dots, 1/\epsilon - 1$, let G_j denote the union of edges E_i for all $i \equiv j$ (modulo $(1/\epsilon)$). As each edge lies in at most one set G_j , there exists some index j such that the sum of $|c_{uv}|$ of all edges in G_j is at most ϵC . Remove all the edges in G_j from the graph G . This decomposes G into a disjoint collection of $(1/\epsilon)$ -outerplanar graphs F_1, F_2, \dots . We solve the problem separately on each of these problems. Since we only remove sets of edges (and no vertices) the solution of the subgraphs gives an assignment for the vertices of the original graph G .

Now consider the quality of the solution. Since the edges in G_j that were removed can contribute at most $-\sum_{(u,v) \in G_j} |c_{uv}|$, the optimal assignment for the subproblems has value at most $H_{\text{opt}} + \epsilon C$. Moreover since we ignore these edges completely while obtaining the block solution, when we consider the value of this assignment for the overall graph, in the worst case these could contribute an additional ϵC to the Hamiltonian. Thus, the assignment obtained by our algorithm has value at most $H_{\text{opt}} + 2\epsilon C$ which by Theorem 5 is at most $H_{\text{opt}}(1 - 6\epsilon)$. Note that the running time of the algorithm is dominated by solving the problem on outerplanar graphs which requires a total time of $O(n2^{3/\epsilon})$. Choosing $\epsilon = 6\delta$ this implies that the algorithm obtains an assignment with value at most $(1 - \delta)H_{\text{opt}}$ in time $O(n2^{18/\delta})$.

SB and BMT acknowledge support by NSA and ARDA through ARO contract number W911NF-04-C-0098.

-
- [1] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution. 2000, <http://arxiv.org/abs/quant-ph/0001106>.
- [2] A. Ambainis and O. Regev. An elementary proof of the quantum adiabatic theorem. 2004, <http://arxiv.org/abs/quant-ph/0411152>.
- [3] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [4] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A numerical study of the performance of a quantum adiabatic evolution algorithm for satisfiability. 2000, <http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0007071>.
- [5] A. Wigderson. P, NP and Mathematics – a computational complexity perspective. <http://www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/W06/W06.pdf>.
- [6] L. Bieche, R. Maynard, R. Rammal, and J.P. Uhry. On the ground states of the frustration model of a spin glass by a matching method of graph theory. *J. Phys. A*, 13:2553–2576, 1980.
- [7] F. Barahona. On the computational complexity of Ising spin glass models. *Jour. of Phys. A: Math. and Gen.*, 15:3241–3253, 1982.
- [8] G. Rose. <http://dwave.wordpress.com/2007/01/19/quantum-computing-demo-announcement/>.
- [9] V.V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, Germany, 2001.
- [10] Sanjeev Arora, Eli Berger, Guy Kindler, Muli Safra, and Elad Hazan. On non-approximability for quadratic programs. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 206–215, Washington, DC, USA, 2005. IEEE Computer Society.
- [11] Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for ising model (extended abstract). In *Automata, Languages and Programming*, pages 462–475, 1990.
- [12] Paola Alimonti and Viggo Kann. Some APX-completeness results for cubic graphs. *Theor. Comput. Sci.*, 237(1-2):123–134, 2000.
- [13] N. Bansal, S. Bravyi, and B.M. Terhal. Classical approximation algorithms for local Hamiltonian problems on planar graphs. In preparation.
- [14] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9:615–627, 1980.
- [15] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41:153–180, 1994.
- [16] Sanjeev Khanna and Rajeev Motwani. Towards a syntactic characterization of PTAS. In *STOC '96: Proceedings of the 28th annual ACM symposium on theory of computing*, pages 329–337, 1996.
- [17] C. Dürr and P. Hoyer. A quantum algorithm for finding the minimum. 1996, <http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/9607014>.
- [18] P.G. Kwiat, J.P. Mitchell, P.D.D. Schwindt, and A.G. White. Grover's search algorithm: An optical approach. *J. Mod. Opt.*, 47:257–266, 2000, <http://arxiv.org/abs/quant-ph/9905086>.
- [19] N. Bhattacharya, H. B. van Linden van den Heuvell, and R. J. C. Spreeuw. Implementation of quantum search algorithm using classical Fourier optics. *Physical Review Letters*, 88:137901, 2002, <http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0110034>.
- [20] N. Robertson and P. D. Seymour. Graph minors II: Algorithmic aspects of treewidth. *Journal of Algorithms*, 7:309–322, 1986.
- [21] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–22, 1993.
- [22] H. L. Bodlaender. Some classes of graphs with bounded treewidth. In *Bulletin of the European Association for Theoretical Computer Science*, pages 116–126, 1988.
- [23] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and

- R. Niedermeier. Fixed parameter algorithms for DOMINATING SET and related problems on planar graphs. *Algorithmica*, 33:461–493, 2002.
- [24] J. Hopcroft and R.E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21:549–568, 1974.
- [25] J. Edmonds and E. Johnson. Matchings, Euler tours and the Chinese postman problem. *Math. Programming*, 5:88–124, 1973.
- [26] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, 2003.

PROOF OF THEOREM 5

We first note that it suffices to show that $\min_S Q(S) \leq -C/3$ since Claim 1 shows that $\min_S H(S) \leq \min_S Q(S)$. Thus we consider planar graphs with only quadratic terms in $H(S)$, i.e. we assume that $d_u = 0$ for all u . Recall that Bieche et al. [6] has shown that this problem can be solved exactly in polynomial time. Our proof of Theorem 5 builds the ideas of Bieche and hence we first describe their ideas. We begin with some notation.

A graph is planar if it can be drawn in the plane such that no edges cross. This drawing defines disjoint regions in the plane that are called faces. A cycle in a graph is a collection of edges $(u_1, u_2), (u_2, u_3), \dots, (u_{\ell-1}, u_\ell)$ where $u_\ell = u_1$. Given two cycles C_1 and C_2 , their sum $C_1 \oplus C_2$ is defined as the symmetric difference of C_1 and C_2 . The faces of planar graph form a cycle basis, that is, every cycle can be expressed as a sum of faces. Given an assignment S , an edge (u, v) is called unsatisfied if u and v are not assigned according the sign of c_{uv} (i.e. if $c_{uv} \geq 0$ but $S_u S_v = 1$ or if $c_{uv} < 0$ but $S_u S_v = -1$). A face F is called frustrated if it contains an odd number of edges with positive weight. A key observation is that for any assignment, a frustrated face must always contain an odd number (hence at least one) of unsatisfied edges. Conversely, if J is a subset of edges such that each frustrated (resp. non-frustrated) face contains exactly an odd (resp. even) number of edges in J , then there is an assignment S such that the unsatisfied edges are exactly those in J . Thus, $Q(S) = -C + 2 \sum_{(u,v) \in J} |c_{uv}|$. Bieche et al. [6] showed that for planar graphs finding such a set J with minimum weight is equivalent to finding the minimum weight T -join in the dual graph of G^* (these terms are defined below).

For a planar graph G , its dual graph G^* is defined as follows: G^* has a vertex for each face in G . Vertices u and v in G^* are connected by an edge if and only if the faces corresponding to u and v in G share a common edge. Given a G , the dual G^* is not necessarily unique (it depends on the drawing of G), however $G^{**} = G$. A subset of edges E' is called a cut-set if removing them disconnects the graph into two or more components. For planar graphs, every cut-set in the dual graph G^* corresponds to a cycle in G . Let $G = (V, E)$ be a graph with edge weights, and let T be a subset of vertices $T \subseteq V$ such that

$|T|$ is even. A T -join is a collection of edges J such that each vertex in T is adjacent to an odd number of edges in J and each vertex in $V \setminus T$ is adjacent to an even number of edges in J . The minimum weighted T -join problem is to find a T -join with minimum weight, and this can be found in polynomial time using matchings. As we mentioned above finding the optimal assignment is equivalent to finding the minimum weight T -join in G^* where T to be the set of vertices corresponding to the frustrated faces in G , and where an edge corresponding to (u, v) in G has weight $|c_{uv}|$.

We will use a polyhedral description of T -joins. For a subset of edges J , let $v(J)$ denote the corresponding $|E|$ -dimensional incidence vector (with 1 in the i -th coordinate if edge i lies in J and 0 otherwise). For a subset of vertices W , let $\delta(W)$ denote the set of edges with one end point in W and other in $V \setminus W$. Given a graph G and the set T , we say that a subset of edges J is an upper T -join if some subset J' of J is a T -join for G . Let P be the convex hull of all vectors $v(J)$ corresponding to the incidence vector of upper T -joins. P is called the up-polyhedra of T -joins. Edmonds and Johnson [25] gave the following exact description of P (see the book by Schrijver [26], Chapter 29, par. 1-6 for further details).

$$\sum_{e \in \delta(W)} x(e) \geq 1, \text{ for all sets } W \text{ s.t. } |W \cap T| \text{ is odd,} \quad (5)$$

$$0 \leq x(e) \leq 1 \quad \text{for all edges } e. \quad (6)$$

This implies that any feasible solution x to the system of inequalities above can be written as a convex combination of upper T -joins, i.e. $x = \sum \alpha_i v(J_i)$ where $0 \leq \alpha_i \leq 1$ and $\sum_i \alpha_i = 1$. In particular this implies that

Corollary 7. *If all the edge weights $w(e)$ are non-negative, then given any feasible assignment $x(e)$ satisfying the inequalities above, there exists a T -join with cost at most $\sum_e w(e)x(e)$.*

We are now ready to prove Theorem 5.

Lemma 8. *Let G be a simple (with no multiple edges between the same pair of vertices) planar graph with weights $|c_{uv}|$ on the edges, and let G^* be its dual graph. For any subset of vertices T of G^* such that $|T|$ is even, the minimum weighted T -join has weight at most $(\sum_{(u,v)} |c_{uv}|)/3$.*

Proof. Each cut-set J of G^* corresponds to a cycle in G . Since G is simple, each cycle has length at least 3, and hence each cut-set J of G^* contains at least 3 edges. Consider the assignment $x(e) = 1/3$. It clearly satisfies Eq. (6). Moreover it also satisfies Eq. (5) as $\delta(W) \geq 3$ for all $W \subset V$, $W \neq V$. The result then follows from Corollary 7. \square

Lemma 8 implies Theorem 5 immediately, since the value of the optimal assignment is $-C$ plus twice the

weight of the optimal T -join which is at most $-C + 2C/3 = -C/3$. Observe that Theorem 5 is tight, as seen from the example where G is a triangle with edge weights $+1, +1$ and -1 . Here $C = 3$, but the optimal spin assignment has value -1 . The condition that G is simple is

necessary. Otherwise, consider the graph on two vertices with two edges, one with weight -1 and other with weight $+1$. Here $C = 2$, but the optimal assignment has value 0 .