The Quantum and Classical Complexity of Translationally Invariant Tiling and Hamiltonian Problems

Daniel Gottesman
Perimeter Institute for Theoretical Physics
Waterloo, Ontario, Canada
dgottesman@perimeterinstitute.ca

Sandy Irani
Computer Science Department
University of California, Irvine
Irvine, CA, USA
irani@ics.uci.edu

Abstract— We study the complexity of a class of problems involving satisfying constraints which remain the same under translations in one or more spatial directions. In this paper, we show hardness of a classical tiling problem on an $N \times N$ 2-dimensional grid and a quantum problem involving finding the ground state energy of a 1-dimensional quantum system of N particles. In both cases, the only input is N, provided in binary. We show that the classical problem is NEXP-complete and the quantum problem is $\mathsf{QMA}_{\mathsf{EXP}}$-complete. Thus, an algorithm for these problems which runs in time polynomial in N (exponential in the input size) would imply that $\mathsf{EXP} = \mathsf{NEXP}$ or $\mathsf{BQEXP} = \mathsf{QMA}_{\mathsf{EXP}}$, respectively. Although tiling in general is already known to be NEXP-complete, to our knowledge, all previous reductions require that either the set of tiles and their constraints or some varying boundary conditions be given as part of the input. In the problem considered here, these are fixed, constant-sized parameters of the problem. Instead, the problem instance is encoded solely in the size of the system.

1. Introduction

One perennial difficulty with practical applications of hardness results is that the practically interesting instances of a hard language may not themselves form a hard class. One approach to solving this problem is the difficult theory of average-case complexity [14], [2], in which one can show that "typical" cases of some language are hard. In this paper we take a different approach. In many cases, practically interesting instances possess some shared property, such as a symmetry, that distinguish them from the general instance and might, in principle, make those instances easier. We will study such an example and show that, even in a system possessing a great deal of symmetry, it is still possible to prove a hardness result.

Specifically, we consider the related problems of determining whether there is a possible tiling of an r-dimensional grid with some fixed set of classical tiles and of finding the lowest energy state (or *ground state*) of a quantum system involving interactions only between neighboring particles on an r-dimensional grid. The ground state energy of a system is considered one of the basic properties of a physical system, and over the last few decades, physicists have developed a number of heuristics that have been successful in finding the ground state energy in many special cases. On the other hand, in earlier work [1], we have shown that in the most general case, even in a 1-dimensional quantum system, finding the ground state is a computationally difficult problem (modulo the usual complexity-theoretic assumptions). However, the construction presented in [1] involves a system which is completely unnatural from a physical point of view. The most interesting physical systems frequently possess an additional symmetry: translational invariance. In this paper, we will show that even a 1-dimensional translationally-invariant system can be hard.

One interesting feature of our proof which may have more general applicability is that the only free parameter for the language we consider is the size of the system. This is frequently the case for interesting systems: there is a basic set of rules of constant size, and we wish to study the effect of those rules when the system to which the rules apply becomes large. In practice, many such systems seem difficult to solve, but it is hard to see how to prove a complexity-theoretic hardness result, since that requires reducing a general problem in some complexity class to the language under consideration, and there doesn't seem to be room in this language to fit all the needed instances. Usually, this difficulty is circumvented by modifying the problem slightly, to add additional parameters in which we can encode the description of the instance we wish to simulate.

To illustrate, let us present the classical tiling problem we study in this paper: We are given a set of square tiles which come in a variety of colors. The area to be tiled is a square area whose size is an integer multiple of the length of a tile. We are given horizontal constraints indicating which pairs of colors can be placed next to each other in the horizontal direction and another set of constraints in the vertical direction. We specify a particular color which must go in the four corners of the grid. The description of the tile colors, placement constraints and boundary conditions are fixed for all inputs of the problems. The input is just a number N written in binary and we wish to know whether an $N \times N$ grid can be properly tiled given these constraints. We show that this problem is NEXP-complete. Note that the input in this case is size $\log N$, so an algorithm to solve our tiling problem that runs in time polynomial in N would imply that $\mathsf{NEXP} = \mathsf{EXP}$. While it is possible that $\mathsf{P} \neq \mathsf{NP}$ and yet $\mathsf{NEXP} = \mathsf{EXP}$, this seems unlikely to be the case.

This version of tiling is equivalent to the more common Wang Tiles [21] in that any set of tiles can be transformed into a set of Wang Tiles (and vice versa) such that there is a one-to-one correspondence between valid tilings on an $N \times N$ grid. For an *infinite* grid, the problem is undecidable. Intuitively, it makes sense that it is also hard (for some sets of tiles) for a finite grid, since there are exponentially many possible tilings, and it is impossible to tell locally whether a given partial tiling can be extended indefinitely. Indeed, there are prior results showing that related tiling problems are NEXP-complete, but to our knowledge, all previous reductions require that either the set of tiles and their constraints or some varying boundary conditions be given as part of the input [7], [15], [5]. For instance, one may specify the placement of some number of tiles and ask whether it is possible to extend that partial tiling to a tiling of the full square. Even though that problem had been proven hard, the more natural problem of whether it is possible to efficiently find a tiling of the empty grid remained open.

Many NEXP-complete problems are succinct versions of familiar combinatorial problems [8], [20] in which the input has some special structure which allows for a more compact representation. For example, consider the problem of finding an independent set in a graph where the graph is specified by indicating the number of nodes in binary and providing a compact rule (or circuit) to determine if two nodes are connected. Traditionally, the rule is included as part of the input, which potentially allows for a more expressive language. The analog to our work would be for the rule to have a constant-sized description, fixed for all inputs.

The basic idea of our construction is to reduce from an instance x of some language in NEXP by encoding x in the binary expansion of N, the size of the grid. It is well known that a finite set of tiling rules can be used to implement a universal Turing Machine. We need some way to express the program for the Turing Machine to run, and that program must grow with the size of x. Previous constructions managed this by resorting to either polylog N different tile types or varying boundary conditions to encode x, but those are both fixed, constant-sized parameters in our version of the problem. Instead, we use the tiles to implement a binary counter which converts N into binary and then uses it as an input to a universal Turing Machine.

The other problem we consider is finding the ground state energy of a quantum system. The state of a quantum system with N qubits is a vector in a Hilbert space of dimension 2^N. We will be considering a slightly more general version in which an individual particle has its state in a space of dimension d, in which case the state of a system of N such particles is a vector in a d^N-dimensional Hilbert space. Any physical property of a system that can be measured (e.g. location, momentum, energy) corresponds to a linear operator. For an N-particle system, it can be expressed as a $d^N \times d^N$ matrix over the complex numbers. If the property is measured, then the outcome must be an eigenvalue of the corresponding linear operator and the state of the system after the measurement is in the eigenspace corresponding to the outcome. Thus, the problem of finding the energy for the lowest energy state is the same as determining the lowest eigenvalue for the energy operator (also called the *Hamiltonian* for the system). The difficulty, of course, is that the Hamiltonian matrix is exponentially large in the size N of the system.

We are typically interested in systems whose Hamiltonians are *local* in that they can be expressed as a sum of terms each of which acts non-trivially only on a constant-sized subset of the particles in the system. Although the term "local" does not imply anything about the physical location of the particles, it is motivated by the idea that particles only interact when they are physically close to each other. We can extend this even further and consider a system of particles on an r-dimensional grid, where the terms of the Hamiltonian operate only on neighboring pairs of particles in the grid. Note that although the full matrix representation of a Hamiltonian is exponentially large in the size of the system, a local Hamiltonian has a compact representation: each term can be expressed as a constant-sized matrix, and there can only be polynomially many such terms.

Kitaev introduced the class QMA, the quantum analog of NP, and showed that the problem of determining the ground state energy of a system defined by a local Hamiltonian is QMA-hard [13]. Thus, we do not hope to solve it even on a quantum computer. With an additional promise, the problem is QMA-complete: there exist two values $a > b$, such that $a - b \geq 1/\mathrm{poly}(N)$, where it is guaranteed that the ground state energy is at most b or at least a, and one wants to determine only which of the two alternatives holds. The problem is still hard even for two-dimensional systems on qubits or one-dimensional systems of particles of constant Hilbert space dimension [18], [1].

Despite these worst-case results, numerical methods have been successful at determining ground state energies for many quantum systems, especially in one dimension. What are the differences between these hard QMA-complete problems and the more tractable systems studied by numerical physicists? One feature of the QMA-completeness constructions is that the individual terms of the Hamiltonian are position-dependent. Essentially, the computation performed by a quantum verifier circuit is encoded into the Hamiltonian so that a low energy state exists if and only if there is a quantum witness that causes a verifier to accept. Thus, the terms of the Hamiltonian encode, among other things, individual gates in a quantum circuit. In contrast, many quantum systems of physical interest are much more uniform in that they consist of a single Hamiltonian term that is simultaneously applied to each pair of neighboring particles along a particular dimension. Such a system is called *translationally invariant*.

Since highly symmetric systems are rather natural, a number of researchers have studied the computational power of translationally invariant quantum systems. For instance, [17] gives a 20-state translation-invariant modification of the construction from [1] (improving on a 56-state construction by [11]) that can be used for universal 1-dimensional adiabatic computation. These modifications require that the system be initialized to a particular configuration in which each particle is in a state that encodes some additional information. The terms of the Hamiltonian, although identical, act differently on different particles depending on their state. The ground state is therefore degenerate and one determines which ground state is reached by ensuring that the system starts in a particular state. Kay [12] gives a construction showing that determining the ground state energy of a one dimensional nearest-neighbor Hamiltonian is QMA-complete even with all two-particle terms identical. The construction does, however, require position-dependent one-particle terms. Irani has demonstrated ground state complexity in one-dimensional translationally-invariant systems by showing that such systems can have ground states with a high degree of quantum entanglement [10]. While quantum entanglement is closely related to the performance of numerical heuristics in practice, the particular states in this construction are easy to compute.

In contrast, we show that there exist 1-dimensional translationally-invariant quantum systems with nearest-neighbor interactions for which finding the ground state energy is complete for $\mathsf{QMA}_{\mathsf{EXP}}$, a quantum analogue of NEXP. As with the classical result, the only parameter which varies in the language is N, the number of particles, and we must use N to encode the instance from which we wish to reduce. The quantum result uses a similar idea to the classical result: we arrange for a control particle to shuttle between the ends of the system and count the number of particles. The binary encoding for the number of particles is then used as an input to a quantum Turing Machine.

It is worth noting that the one-dimensional version of the classical tiling problem is very easy: it is in P (see the extended version [9] for the algorithm). That is, it can be solved in a time polylog N, whereas it appears the quantum problem can not be done in time poly(N), even on a quantum computer (unless $\mathsf{QMA}_{\mathsf{EXP}} = \mathsf{BQEXP}$, where BQEXP is like BQP, but with exponential circuits). Translational invariance does seem to simplify the 1-dimensional classical case, reducing poly(N) time to polylog(N) time, but it doesn't help very much in the quantum case.

Note that the classical tiling problem is a special case of the ground state energy problem for quantum systems where the Hamiltonian is diagonal in the standard basis with only 1 or 0 entries. Any ground state of such a system is a classical state in which the state of each particle is specified by one of the d possible standard basis states, which correspond to the possible tile colors. A pair of tiles (t_i, t_j) is allowed by the tiling rules iff the corresponding $|t_i t_j\rangle\langle t_i t_j|$ term of the Hamiltonian is 0, so that allowed tilings have 0 total energy, whereas a forbidden tiling has energy at least 1.

2. Problems and Results

Definition 2.1: **TILING**

Problem Parameters: A set of tiles $T = \{t_1, \ldots, t_m\}$. A set of horizontal constraints $H \subseteq T \times T$ such that if t_i is placed to the left of t_j, then it must be the case that $(t_i, t_j) \in H$. A set of vertical constraints $V \subseteq T \times T$ such that if t_i is placed below t_j, then it must be the case that $(t_i, t_j) \in V$. A designated tile t_1 that must be placed in the four corners of the grid.

Problem Input: Integer N, specified in binary.

Output: Determine whether there is a valid tiling of an $N \times N$ grid.

Theorem 2.2: TILING is NEXP-complete.

We give the proof in section 3. The basic idea is that two adjacent corners are used to create a border around the perimeter of the grid which allows us to implement special rules at the top and bottom rows. The interior of the grid is tiled in two layers, each of which implements the action of a Turing machine. The first TM proceeds from top to bottom on layer 1 and the second proceeds from bottom to top on layer 2. The first TM takes no input and acts as a binary counter for N steps. The bottom row of the first layer then holds a binary number that is $\Theta(N^{1/k})$. The rules for the lower boundary are then used to copy the output from the binary counter to the bottom row of layer 2, which acts as the input to a generic non-deterministic Turing machine. The rules for the top boundary check whether the final configuration on layer 2 is an accepting state.

Note that it is important that we chose to have the input N provided in binary. If it were instead given in unary, there would only be one instance per problem size, and the problem would be trivially in P/poly. Thus, in order to prove a meaningful hardness result, we are forced to move up the exponential hierarchy and prove the problem is NEXP-complete rather than NP-complete.

A common convention for this tiling problem is to only specify the boundary condition tile in a single corner of the grid. This does not work in our case: If only the upper right corner tile is specified, a valid tiling for an $N \times N$ grid can be cropped by removing the leftmost column and bottommost row to give a valid tiling for the $(N-1) \times (N-1)$ grid. Thus, there exists $N_0 \in \mathbb{Z}^+ \cup \{\infty\}$ such that if $N < N_0$, there is a valid tiling, and if $N \geq N_0$, then there is no valid tiling. The resulting language either consists of all strings or all strings up to a fixed string in the lexicographical ordering of binary strings. Because tiling the infinite plane is undecidable, N_0 is uncomputable as a function of (T, H, V). Still, if we fix the tiling rules, we know there exists a straightforward algorithm to solve this variant of TILING: simply determine if $N < N_0$. We just do not know N_0, so we do not know precisely what algorithm to use.

Instead of setting a single boundary condition tile, we use specified tiles in all four corners to mark out the boundary of the grid to be tiled. We have considered other versions of the classical translationally-invariant tiling problem to understand to what extent the precise definition of the problem is important. The boundary conditions, as noted above, are a critical component. As well as fixing the tiles at the 4 corners of the square, we have considered periodic boundary conditions (so we are actually tiling a torus) and open boundary conditions, where any tile is allowed at the edges of the square. The case of periodic boundary conditions is particularly interesting because it is truly translationally invariant, unlike our usual formulation where the boundaries break the translational symmetry.

Another variant is to make the problem more similar to the quantum Hamiltonian problem by assigning a cost to any pair of adjacent tiles, and allowing the costs to be different from 0 or 1. This is like a weighted version of tiling and corresponds to a Hamiltonian which is diagonal in the standard basis but does not have any other constraints.

Definition 2.3: **WEIGHTED TILING**
Problem Parameters: A set of tiles $T = \{t_1, \dots, t_m\}$. A set of horizontal weights $w_H : T \times T \to \mathbb{Z}$, such that if t_i is placed to the left of t_j, there is a contribution of $w_H(t_i, t_j)$ to the total cost of the tiling. A set of vertical weights $w_V : T \times T \to \mathbb{Z}$, such that if t_i is placed below t_j, there is a contribution of $w_V(t_i, t_j)$ to the total cost of the tiling. A polynomial p. Boundary conditions (a tile to be placed at all four corners, open boundary conditions, or periodic boundary conditions).
Problem Input: Integer N, specified in binary.
Output: Determine whether there is a tiling of an $N \times N$ grid such that the total cost is at most $p(N)$.

We have also considered problems with additional symmetry beyond the translational invariance. If we have *reflection symmetry*, then if $(t_i, t_j) \in H$, then $(t_j, t_i) \in H$ as well, and if $(t_i, t_j) \in V$, then $(t_j, t_i) \in V$ also. That is, the tiling constraints to the left and right are the same, as are the constraints above and below. However, if we only have reflection symmetry, there can still be a difference between the horizontal and vertical directions. If we have *rotation symmetry*, we have reflection symmetry and also $(t_i, t_j) \in H$ iff $(t_i, t_j) \in V$. Now the direction does not matter either. These additional symmetries are well motivated from a physical point of view since many physical systems exhibit reflection and rotation symmetry. Finally, we have studied the one-dimensional version of the problem as well as the two-dimensional version. See Table 1 for a summary of our results. Proofs are given in the extended version [9], and we sketch the main ideas in section 5.

As noted above, TILING with open boundary conditions is easy but in a strange non-constructive sense in that the efficient algorithm depends on an uncomputable parameter N_0. This case is denoted as "P, uncomputable" in Table 1 (with a question mark for other cases for which we have not been able to prove whether N_0 is computable or not). Note that this does not exclude the existence of a (potentially slower) algorithm to solve particular instances; indeed, all the classes in Table 1 are included in NEXP. For these variants, we know that there is an efficient algorithm, so a hardness result can be ruled out, but since the algorithm depends on an uncomputable parameter, it may be that the problem remains hard in practice.

Now we turn to the quantum problem. First we need to define the class $\mathsf{QMA}_{\mathsf{EXP}}$. It will be a bit more convenient to work with quantum Turing Machines than quantum circuits. The definition is the same as QMA except that the witness and the length of the computation for the verifier (which is a quantum Turing Machine) can be of size 2^{n^k} on an input of length n.

Definition 2.4: A language L is in $\mathsf{QMA}_{\mathsf{EXP}}$ iff there exists a k and a Quantum Turing Machine M such that for each instance x and any $|\psi\rangle$ on $O(2^{|x|^k})$ qubits, on input $(x, |\psi\rangle)$, M halts in $O(2^{|x|^k})$ steps. Furthermore, (a) if $x \in L_{\text{yes}}$, $\exists |\psi\rangle$ such that M accepts $(x, |\psi\rangle)$ with probability at least $2/3$. (b) if $x \in L_{\text{no}}$, then $\forall |\psi\rangle$, M accepts $(x, |\psi\rangle)$ with probability at most $1/3$.

Definition 2.5: **r-DIM TIH (Translationally-Invariant Hamiltonian)**
Problem Parameter: r Hamiltonian terms $H_1, \dots, H_r$ that each operate on two finite dimensional particles, specified with a constant number of bits. Two polynomials p and q.
Problem Input: Integer N, specified in binary.
Promise: Consider an N^r-dimensional grid of particles and the Hamiltonian resulting from applying H_i to each pair of neighboring particles along dimension i. The ground state energy of this system is either at most $p(N)$ or at least $p(N) + 1/q(N)$.
Output: Determine whether the ground state energy of the system is at most $p(N)$ or at least $p(N) + 1/q(N)$.

The following theorem is the main result for the quantum case and shows that the problem will likely be, in general, difficult. Note that typically, one is willing to spend time that is polynomial in the size of the system (which is in turn exponential in the size of the input). It follows from the result that if there is a quantum algorithm that finds the ground state energy in time that is polynomial in the size of the system then $\mathsf{QMA}_{\mathsf{EXP}} = \mathsf{BQEXP}$.

Theorem 2.6: 1-DIM TIH is $\mathsf{QMA}_{\mathsf{EXP}}$-complete.

The theorem immediately implies that r-DIM TIH is $\mathsf{QMA}_{\mathsf{EXP}}$-complete for any $r \geq 1$ since we can always take $H_i = 0$ for $i \geq 2$ which results in a system of N^{r-1} independent lines with N particles. We sketch the proof of theorem 2.6 in section 4.

We can also consider variants of 1-DIM TIH. If we

	2-D, no symmetry	2-D, reflection sym.	2-D, rotation sym.	1-D
BC on all corners				
unweighted	NEXP-complete	P, uncomputable?	P	P
weighted	NEXP-complete	NEXP-complete	P	P
BC on 0 or 1 corner				
unweighted	P, uncomputable	P	P	P
weighted	NEXP-complete	NEXP-complete	P	P
Periodic BC				
unweighted	NEXP-complete*	P, uncomputable?	P	P
weighted	NEXP-complete	NEXP-complete	P	P

Figure 1. Summary of the variants of TILING. "BC" is short for "boundary condition." "P, uncomputable" means that the associated problem is in P, but an essential parameter of the efficient algorithm we found is uncomputable (with a question mark if we are not sure whether it is uncomputable). "NEXP-complete*" means complete under an expected poly-time randomized reduction or a deterministic polyspace reduction.

use periodic boundary conditions instead of open boundary conditions, we get the same result. If we add reflection symmetry, the problem also remains $\mathsf{QMA}_{\mathsf{EXP}}$-complete with open or periodic boundary conditions. See the extended version [9] for proof of these results.

As is common in QMA-completeness results, the construction for Theorem 2.6 creates a Hamiltonian whose ground state is a uniform superposition of a sequence of states which represent a particular process. A portion of the Hilbert space for the system holds a clock which allows us to control the length of the sequence and ensures that the states in the sequence are mutually orthogonal. That is, the t^{th} state has the form $|\phi_t\rangle|t\rangle$, where $|\phi_t\rangle$ is the t^{th} state in the process we wish to simulate, and the overall ground state will be $\sum_t |\phi_t\rangle|t\rangle$. The size of the system controls the number of time steps for which the clock runs. In the case of the construction presented here, the process consists of two main phases. The first phase is the execution of a Turing machine which simply increments a binary counter. The clock ensures that this TM is run for $N-2$ steps after which a number that is $\Theta(N^{1/k})$ is encoded in binary in the state of the quantum system. This state is then used as the input to an arbitrary quantum Turing machine which is executed in the second phase. This QTM implements a verifier which is also allowed a quantum witness of length $\Theta(N)$. Finally, there is an energy term which penalizes any non-accepting computation of the verifier.

As a corollary of Theorem 2.6, the following version of N-REPRESENTABILITY [16] is also $\mathsf{QMA}_{\mathsf{EXP}}$-complete: Given a density matrix ρ on two d-state particles, is it within ϵ of a state ρ' such that there exists a translationally-invariant pure state $|\psi\rangle$ for N particles arranged in a circle for which ρ' is the marginal state of two adjacent particles? ρ is a parameter of the problem, and N, given in binary, is the only input, as in our Hamiltonian problem. We can reduce to this version of N-REPRESENTABILITY by starting with 1-DIM TIH on a circle. Then there is always a translationally-invariant pure ground state $|\psi\rangle$ of the Hamiltonian H. By breaking the Hilbert space of two d-state particles up into small balls, we can get a finite set of density matrices ρ to try. For each one, if we can solve N-REPRESENTABILITY, we can determine if ρ can be extended to a candidate ground state $|\phi\rangle$, and if so we can determine the energy of $|\phi\rangle$, since it is just equal to $N\mathrm{tr}(H_1\rho)$. Trying all possible ρ, we can thus find the ground state energy of H, up to some ϵ-dependent precision.

3. Hardness of TILING

The construction will make use of a binary counter Turing machine M_{BC} which starts with a blank semi-infinite tape. The head begins in a designated start state in the left-most position of the tape. M_{BC} will generate all binary strings in lexicographic order. More specifically, there is a function $f : \mathbb{Z} \rightarrow \{0,1\}^*$ such that for some constant N_0 and every $N \geq N_0$, if M_{BC} runs for N steps, then the string $f_{BC}(N)$ will be written on the tape with the rest of the tape blank. Moreover there are constants c_1 and c_2 such that if n is the length of the string $f_{BC}(N)$ and $N \geq N_0$, then $2^{c_1 n} \leq N \leq 2^{c_2 n}$. We will also assume that for any binary string x, we can compute N such that $f_{BC}(N) = x$ in time that is polynomial in the length of x. In some of the variations of the problem we consider we will need to put additional restrictions on N (such as requiring N to be odd), and in those cases, we still require that we can find an N with the appropriate restrictions such that $f_{BC}(N) = x$.

Using a standard padding argument, we can reduce any language in NEXP to $\mathsf{NTIME}(2^{c_1 n})$. If L is in $\mathsf{NTIME}(2^{n^k})$, the reduction consists of padding an input x so that its length is $|x|^k/c_1$ [19]. Thus, we will take an arbitrary non-deterministic Turing machine M which accepts a language L in time $2^{c_1 n}$ and reduce it to TILING. The tiling rules and boundary conditions will be specific to the Turing machine M but will be independent of any particular input. The reduction for Theorem 2.2 then will take an input string x and output integer N such that $f_{BC}(N-3) = x$. The tiling rules will have the property that a string x is in L if and only if an $N \times N$ grid can be tiled according to the tiling rules.

Proof of Theorem 2.2. The boundary conditions for the $N \times N$ grid will be that the four corners of the grid must have

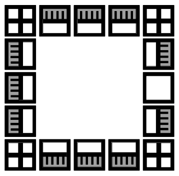
Figure 2. A possible tiling of the sides of a 5×5 grid.

a designated tile type ⊞. (We will actually only need to use the upper left and bottom left corners.) First we will specify a set of boundary tiles and their constraints. In addition to ⊞ there are four other kinds of boundary tiles: ◧, ⬒, ⬓, ◨. We will call the rest of the tiles *interior* tiles. ◧ will mark the left side of the grid, ⬒ the top of the grid, ⬓ the bottom of the grid, and ◨ the right side of the grid. (See figure 2.)

Nothing can go to the left of a ◧ tile which means that the only place a ◧ tile could go is the left-most boundary, as desired. Similarly, nothing can above a ⬒ tile, nothing can go below a ⬓ tile, and nothing can go to the right of a ◨ tile, which means ⬒ tiles can only go in the top row, ⬓ tiles can only go in the bottom row, and ◨ tiles can only go in the right-most column. No interior tile can border a ⊞ tile in any direction. Furthermore a ⊞ cannot border on itself in any direction. This means that the only possible locations for a ⊞ are the four corners since those are the only places which can be surrounded by ◧, ⬒, ⬓, or ◨ tiles. Since the boundary conditions state that ⊞ tiles must go in the corners, those are exactly the locations that will hold ⊞ tiles. The only tiles that can go above or below the ⊞ tiles are ◧ and ◨ tiles. The only tiles that can go either to the left or right of the ⊞ tiles are ⬒ and ⬓ tiles. We will add the constraint that the only tiles that can go above or below ◧ tiles are ◧ tiles or ⊞ tiles. Thus the entire west boundary, except for the corners, will be ◧ tiles. Similarly, we add constraints that ⬓ tiles must have ⊞ or ⬓ tiles to their right and left and any ⬒ tile must have a ⊞ or ⬒ tile to its right and left. This means that the entire south border except for the corners is tiled with ⬓ tiles and the entire north border except for the corners will be ⬒ tiles. We do not need to put any further restrictions on the right boundary.

The remainder of the grid will be tiled in two layers. The constraints on the two layers only interact at the bottom of the grid, so we describe each layer separately. The actual type for an interior tile is specified by a pair denoting its layer 1 type and layer 2 type. The bottom layer will be used to simulate the Turing machine M_{BC}. The top boundary of the grid will be used to ensure that M_{BC} begins with the proper initial conditions. Then the rules will enforce that each row of the tiling going downwards advances the Turing machine M_{BC} by one step. At the bottom of the grid, the output is copied onto layer 2. Layer 2 is then used to simulate a generic non-deterministic Turing machine on the input copied from layer 1. The lower left corner is used to initialize the state of M and the constraints enforce that each row going upwards advances the Turing machine M by one step. Finally, the only states of M that are allowed to be below an ⬒ tile are accepting states. Since each Turing machine only executes for $N-3$ steps and the grid has space for $N-2$ tape symbols, the right end of the tape will never be reached.

Although it is well known that tiling rules are Turing complete [3], we review the ideas here in order to specify the details in our construction. We will assume that the Turing machine M is encoded in a tiling that goes from bottom to top. This can easily be reversed for M_{BC} which goes from top to bottom. The non-deterministic Turing machine M is specified by a triplet (Σ, Q, δ), with designated blank symbol $\# \in \Sigma$, start state $q_0 \in Q$ and accept state $q_A \in Q$. There are three varieties of tiles, designated by elements of Σ (variety 1), $\Sigma \times Q$ (variety 2) and $\Sigma \times Q \times \{R, L\}$ (variety 3). Variety 1 represents the state of the tape away from the Turing machine head. Variety 2 represents the state of the tape and head when the head has moved on to a location but before it has acted. Variety 3 represents the state of the tape and head after the head has acted, and the $\{R, L\}$ symbol tells us which way the head moved. In the horizontal direction a tile corresponding to $a \in \Sigma$ can go next to any other interior tile to the left or right. Variety 2 types cannot go next to each other nor can variety 3 types. The only allowed pairings of variety 2 tiles and variety 3 tiles in the horizontal direction are of the form: $[a, q]$ to the left of $[b, q, L]$ or $[a, q]$ to the right of $[b, q, R]$. Note that the state q of the head must be the same and the variety 2 tile must be placed in the direction designated by the R or L in the variety 3 tile.

In the vertical direction, for any $a \in \Sigma$ we can have $[a]$ above $[a]$. Variety 2 tiles cannot go above or below each other and variety 3 tiles cannot go above or below each other. A variety 1 or a variety 2 tile can go above a variety 3 or a variety 1 tile as long as the alphabet symbols are the same. That is, $[a, q]$ or $[a]$ can go above either $[a, q', L/R]$ or $[a]$. Finally, a variety 3 tile must go above a variety 2 tile and a variety 2 tile must go below a variety 3 tile. Furthermore, these parings are allowed only if they encode a valid move of the Turing machine. That is, $[b, q', L]$ can go above $[a, q]$ only if $(a, q) \rightarrow (b, q', L)$ is one of the valid non-deterministic moves of the machine. Similarly $[b, q', R]$ can go above $[a, q]$ only if $(a, q) \rightarrow (b, q', R)$ is one of the valid non-deterministic moves of the machine. The table below gives an example of a section of tiles that encodes the move $(a, q) \rightarrow (b, q', L)$ of the Turing machine:

$[c, q']$	$[b, q', L]$	$[d]$
$[c]$	$[a, q]$	$[d, q, L]$

The lower row shows the head in the square with the a. The $[d, q, L]$ is from the previous TM move. The tile

$[b, q', L]$ enforces that the tiler is committing to executing the step $(a, q) \to (b, q', L)$, although there may have been other non-deterministic choices. The $[c, q']$ tile to the left of the $[b, q', L]$ shows the new location and state of the head. The $[b, q', L]$ tile now just acts as a $[b]$ tile for purposes of the tiling above.

For our particular construction, we would like to start out the Turing machine M_{BC} with $[q_0, \#]$ in the leftmost location followed by $[\#]$ tiles. For layer 1, the only tiles that can go below a ⬒ tile are $[q_0, \#]$ or $[\#]$ tiles. We forbid having a $[\#]$ tile to the right of a ◧ tile and we do not allow a $[q_0, \#]$ tile to the right of a $[\#]$ tile. We can assume without loss of generality that the Turing machine overwrites the leftmost # on the tape and never writes a # there again. We can also assume that the Turing machine never transitions back to the q_0 state. The rest of the layer 1 rules just enforce the rules for the Turing machine M_{BC}.

Now in order to copy the output from M_{BC} to the input tape for M, we restrict the kinds of tiles that can go above ⬓ tiles. A layer 2 tile that goes above a ⬓ must be $[a]$ or $[a, q_0]$ for some $a \in \Sigma$. Furthermore, in the space above an ⬓ tile, the alphabet characters for the layer 1 and layer 2 tiles must match. This copies the output of M_{BC} onto the input of V. Now we want to ensure that the starting configuration of V has only one head in the leftmost location. To accomplish this, we forbid a ◧ to go next to an $[a]$ tile for $a \in \Sigma_{M_{BC}}$ and forbid an $[a]$ tile (for all $a \in \Sigma$) to the left of a $[b, q_0]$ tile. Again, we can assume that M never transitions back to q_0. A little care must be taken to overwrite the leftmost input tape character with something that is not in the alphabet of M_{BC}. This is because we have forbidden having an $[a]$ tile to the right of a ◧ for any $a \in \Sigma_{M_{BC}}$. The information encoded in the left-most tape symbol can be retained by having a new a' symbol in Σ_M for every $a \in \Sigma_{M_{BC}}$.

Finally, the only variety 2 tiles on layer 2 which we allow below a ⬒ tile must be of the form $[a, q_A]$, where q_A is the accepting state. Thus, there is a valid tiling if and only if the non-deterministic TM M accepts x in $N - 3$ steps.

4. The Quantum Case

As in the 2-dimensional classical tiling problem, we make use of a binary counting Turing machine M_{BC}. Because we are working with quantum systems, we will require that M_{BC} be reversible. Bernstein and Vazirani [4] have shown that any deterministic Turing machine can be made reversible, meaning that given a configuration of the Turing machine, it has a unique predecessor in the computation. There may be some additional overhead but it is not significant. We can still assume that there is a function $f : \mathbb{Z} \to \{0, 1\}^*$ such that for some constant N_0 and every $N \geq N_0$, if M_{BC} runs for N steps, then the string $f_{BC}(N)$ will be written on the tape with the rest of the tape blank. Moreover there are constants c_1 and c_2 such that if n is the length of the string $f_{BC}(N)$ and $N \geq N_0$, then $2^{c_1 n} \leq N \leq 2^{c_2 n}$. We will also assume that for any binary string x, we can compute N such that $f_{BC}(N - 2) = x$ in time that is polynomial in the length of x.

We can reduce any language in $\mathsf{QMA}_{\mathsf{EXP}}$ to a language L that is accepted by a verifier who uses a witness of size $2^{c_1 n}$ and whose computation lasts for $2^{c_1 n}$ steps, where n is the length of the input. This is the same reduction used in the classical case, in which the input is padded to length $|x|^k / c_1$. We can use standard boosting techniques to assume that the probability of acceptance or rejection is at least $1 - \epsilon$ or at most ϵ for $\epsilon = 1/\mathrm{poly}(N)$ [13]. Suppose we are given an arbitrary verifier quantum Turing machine V which takes as input a classical/quantum pair $(x, |\psi\rangle)$ such that $|\psi\rangle$ has $2^{c_1 n}$ qubits and halts in $2^{c_1 n}$ steps. Based on V, we will produce a Hamiltonian term H which acts on a pair of finite-dimensional particles. We will also produce two polynomials p and q. The reduction for Theorem 2.6 will then take input string x and output an integer N such that $f_{BC}(N-2) = x$. The Hamiltonian will have the property that for any x, if there exists a $|\psi\rangle$ that causes V to accept with probability at least $1-\epsilon$, then when H is applied to every neighboring pair in a chain of length N, the resulting system has a unique ground state whose energy is at most $p(N)$. If for every $|\psi\rangle$, M accepts with probability at most ϵ, then the ground state energy of the system is at least $p(N) + 1/q(N)$.

Quantum Turing machines were first introduced in [6] and further developed in [4]. The latter paper showed that we can make a number of simplifying assumptions about the form of a quantum Turing machine and not restrict its power in a complexity-theoretic sense. In particular, we can assume without loss of generality that the Turing machine V has a one-way infinite tape and that the head starts in designated start state q_0 at the left-most end of the tape. We will also assume that on input x, after $2^{c_1 |x|}$ steps, the Turing machine is in an accepting or rejecting state and the head is again at the left-most end of the tape. We will also assume that the witness will be stored in a parallel track with the left-most qubit in the left-most position of the tape.

We now describe the set of states for the particles. A standard basis state for the whole system will be denoted by the state for each particle. States ⧀ and ⧁ are special bracket states that occur at the ends of the chain.

Definition 4.1: A standard basis state is *bracketed* if the left-most particle is in state ⧀, the right-most particle is in state ⧁, and no other particle in the chain is in state ⧀ or ⧁. $\mathcal{S}_{br}$ is the space spanned by all bracketed states.

We will restrict our attention for now to $\mathcal{S}_{br}$ and add a term later that gives an energy penalty to any state outside $\mathcal{S}_{br}$. The rest of the states will be divided into six tracks, so the state of a particle is an ordered 6-tuple with each entry specifying the state for a particular track. The set of allowable states will not necessarily be the full cross product of the states for each track.

Two of the tracks will implement a clock, with one track working as sort of a second hand and another track as a minute hand. The other four tracks will be used to implement two Turing machines which share a work tape. Track 3 holds the work tape. Track 4 holds the state and head location for the first Turing Machine (which is M_{BC}) and Track 5 holds the state and head location for the second Turing Machine (which is V). The sixth track will hold the quantum witness for V. Since there is limited interaction between the tracks, it will be simpler to describe the Hamiltonian as it acts on each track separately and then describe how they interact. Figure 3 below gives a picture of the start state for the system. Each column represents the state of an individual particle.

As is typical in hardness results for finding ground state energies, the Hamiltonian applied to each pair will consist of a sum of terms. There are two types of terms. Type I terms will have the form $|ab\rangle\langle ab|$ where a and b are possible states. This has the effect of adding an energy penalty to any state which has a particle in state a to the immediate left of a particle in state b. We will say a configuration is *legal* if it does not violate any Type I constraints. Type II terms will have the form: $\frac{1}{2}(|ab\rangle\langle ab| + |cd\rangle\langle cd| - |ab\rangle\langle cd| - |cd\rangle\langle ab|)$. These terms enforce that for any eigenstate with zero energy, if there is a configuration A with two neighboring particles in states a and b, there must be a configuration B with equal amplitude that is the same as A except that a and b are replaced by c and d. Even though a Type II term is symmetric, we associate a direction with it by denoting it with $ab \rightarrow cd$. Type II terms are also referred to as *transition rules*. We will say that configuration A transitions into configuration B by rule $ab \rightarrow cd$ if B can be obtained from A by replacing an occurrence of ab with an occurrence of cd. We say that the transition rule applies to A in the forward direction and applies to B in the backwards direction. We will choose the terms so that for any legal configuration, at most one transition rule applies to it in the forward direction and at most one rule applies in the backwards direction. Thus, a state satisfying all Type I and Type II constraints must consist of an equal superposition of legal configurations such that there is exactly one transition rule that carries each configuration to the next. The illegal pairs are chosen so that any state which satisfies the Type I and Type II constraints corresponds to a process we would like to simulate or encode in the ground state. In our case, the process is the execution of two Turing Machines each for at most $N-2$ steps, where N is the length of the chain.

In the remainder of this section, we give a brief overview of the construction. See the extended version [9] for details.

Illegal pairs are used to enforce that the state of Track 1 is always of the form given by the regular expression $\textcircled{<}\textcircled{\circ}^*(\textcircled{\rightarrow}+\textcircled{\leftarrow})\bigcirc^*\textcircled{>}$. There is one arrow symbol on Track 1 that shuttles back and forth between the left end and the right end and operates as a second hand for our clock. We call one round trip of the arrow on Track 1 an *iteration*. Every iteration has $2(N-2)$ distinct states and $2(N-2)$ transitions. Each iteration causes one change in the configuration on Track 2 which acts then as a minute hand for the clock. The Track 2 states are partitioned into two phases. The first phase is called the *Counting Phase* and consists of all $N-2$ of the states of the form $\textcircled{<}\textcircled{1}^*\textcircled{\bar{0}}\textcircled{0}^*\textcircled{>}$. The second phase is the *Computation Phase* and consists of all $N-2$ of the states of the form $\textcircled{<}\textcircled{1}^*\textcircled{\bar{1}}\textcircled{2}^*\textcircled{>}$ states. The $\textcircled{<}\textcircled{1}^*\textcircled{\bar{0}}\textcircled{0}^*\textcircled{>}$ states are ordered according to the number of particles in state $\textcircled{1}$ and the $\textcircled{<}\textcircled{1}^*\textcircled{\bar{1}}\textcircled{2}^*\textcircled{>}$ states are ordered according to the number of particles in state $\textcircled{2}$. Note that the state immediately after $\textcircled{<}\textcircled{1}^*\textcircled{\bar{0}}\textcircled{>}$ in the ordering is $\textcircled{<}\textcircled{1}^*\textcircled{\bar{1}}\textcircled{>}$. The ground state for the clock is the uniform superposition of all the clock states. We need to have illegal pairs that cause all other states to have an energy cost. As is the case in other such proofs, it is not possible to disallow all states directly with illegal pairs. Instead, we need to show that some states are unfavorable because they evolve via forward or backwards transitions to high energy states.

Each of the arrow states for Track 1 will come in three varieties: $\textcircled{\overset{\rightarrow}{0}}$ and $\textcircled{\overset{\leftarrow}{0}}$ will be used during the initial minute of the clock when it is in state $\textcircled{<}\textcircled{\bar{0}}\textcircled{0}^*\textcircled{>}$ and will be used to check initial conditions on the other tracks. $\textcircled{\overset{\rightarrow}{1}}$ and $\textcircled{\overset{\leftarrow}{1}}$ will be used during the counting phase and $\textcircled{\overset{\rightarrow}{2}}$ and $\textcircled{\overset{\leftarrow}{2}}$ will be used during the computation phase. $\textcircled{\overset{\rightarrow}{1}}$ and $\textcircled{\overset{\rightarrow}{2}}$ will be used to trigger different actions on the other tapes. Every time the $\textcircled{\overset{\rightarrow}{1}}$ sweeps from the left end of the chain to the right end of the chain, it causes M_{BC} to execute one more step. Thus, M_{BC} is run for exactly $N-2$ steps. At the end of the counting phase, Track 3 contains $f_{BC}(N-2)$, which acts as the input for the Turing machine V. The $\textcircled{\overset{\rightarrow}{2}}$ symbol is what causes V to execute a step, so at the end of the computation phase, V has decided whether to accept the witness on Track 6 with input $f_{BC}(N-2)$. We then add a term that penalizes any state which is in the final clock state and does not have an accepting Turing machine state. Thus, only accepting computations will have low energy.

Finally we use an additional term to enforce the boundary conditions. This is achieved by weighting the Hamiltonian terms for the illegal pairs and transition rules by a factor of three. Then an additional term is applied to each particle, which gives a benefit to any particle that is in the $\textcircled{<}$ or $\textcircled{>}$ state. Only the left-most and right-most particles can obtain this energy benefit without incurring the higher cost of having an endpoint state in the middle of the chain.

5. VARIANTS OF TILING AND 1-DIM TIH

Here we briefly summarize the main idea for the variants of TILING and 1-DIM TIH. For the full proofs, see [9]. For the hard cases, the proof generally consists of adding extra layers of tiles that will mimic the four-corners boundary condition and break any extra reflection symmetry of the rules. The easy cases typically take a tiling of some size and modify it to make tilings of larger or smaller sizes.

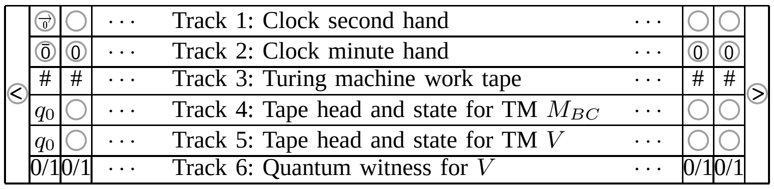
Track 1: Clock second hand
Track 2: Clock minute hand
#
Track 3: Turing machine work tape
q_0
Track 4: Tape head and state for TM M_{BC}
Track 5: Tape head and state for TM V
0/1
Track 6: Quantum witness for V

Figure 3. The start state for the system. Each row represents a track and each column represents the state of an individual particle.

One-D TILING: We consider the graph whose nodes are the different types of tile, with an edge between any two tiles that can be adjacent. The graph is of constant size, so most properties of it can be pre-computed. We want to know whether the graph has a path of exactly length N satisfying the appropriate boundary conditions. For large N, the path can be taken to consist of a constant-length path plus many copies of one cycle, and whether such a path is possible can be easily computed. For WEIGHTED TILING, the graph is a weighted graph and we wish to find a minimal-cost path, which can be done in essentially the same way.

TILING with open BC: See section 2.

TILING with periodic BC: For this case, we restrict N to be an odd prime. We need a randomized reduction to choose an appropriate N. We can create tiling rules which largely alternate between two tile types ■ and □, but since N is odd, the checkerboard pattern cannot wrap all the way around, and there must be vertical and horizontal lines of other types of tiles present as well. These will serve as the borders of a grid within which the regular TILING solution can be implemented. With no further constraints, it would be possible for there to be multiple vertical and horizontal lines, dividing the grid into multiple smaller rectangles, but we add extra tile types that form a diagonal line within each rectangle. This forces the rectangles to be squares and the sides of the squares to divide N. Since N is prime, there can only be a single vertical line and a single horizontal line.

WEIGHTED TILING: For open BC, for each corner, we can add a tile type and corresponding rules so that the new type has a negative cost to be in that corner, but a positive cost elsewhere. These new tiles set the boundary conditions at the corners. For the periodic BC case, we again have a checkerboard and use odd N to ensure there is at least one vertical and one horizontal line, but we make the intersection of the lines have positive cost to ensure that there is not more than one of each type of line.

1**-DIM TIH with periodic BC:** Again we use odd N, and add an extra track with alternating states Ⓐ and Ⓑ. Since N is odd, there must be another type of state ⦶ which we use to define the ends of the line, but we assign a positive cost to ⦶ so that there will only be one such state.

TILING with reflection symmetry: If AB is a legal configuration, so is $ABAB$. Thus, given any valid tiling of an $N \times N$ grid, we can extend it to a valid tiling of an $(N+2) \times (N+2)$ grid. We need only find a single even-N solution and a single odd-N solution to get solutions for all larger N. Whether these solutions exist might be uncomputable for the four corners or periodic boundary conditions. For open boundary conditions an $N \times N$ tiling can be extended or cropped to tile a square of size $N+1$ or $N-1$, so we only need to check if there is a valid tiling of a 2×2 square.

WEIGHTED TILING with reflection symmetry: We use additional layers of tile to break the symmetry. We arrange the costs so that a checkerboard pattern of four types of tile □, ■, ▒, and ▓ is preferred in most places, but another sort of tile prefers to line the edges (for open or four corners BC) or break up the checkerboard with a vertical and a horizontal line (for periodic BC). Only ▓ and □ can border the vertical edge of divider tiles without a large cost penalty, so by choosing the parity of N appropriately, we enforce an interruption in each row with another kind of tile. We can force these additional tiles to form a diagonal line, and by giving them a positive cost, ensure that there is only one in each row and column. Then the tiles above and to the left of the diagonal line will be of different types than the tiles below and to the right. This breaks the reflection symmetry near the diagonal line, and by adding additional layers of tile, we can extend this broken symmetry to the whole grid. The main layer can refer to the extra layers to determine which directions are left and right or up and down.

1**-DIM TIH with reflection symmetry:** For this case, we use the arrow state in Track 1 to break the reflection symmetry. We can choose N odd and almost fill Track 1 with alternating Ⓐ and Ⓑ states. However, we only let Ⓐ go next to the ends of the line, and put a small positive cost to an arrow state, meaning there will be exactly one arrow state in a ground state. When the arrow is in the interior of the track, it has an Ⓐ state on one side and a Ⓑ state on the other, distinguishing left and right. Each time the arrow state moves, the correspondence between Ⓐ and Ⓑ with left and right switches, but we can keep track of that by switching the internal state of the arrow. Track 2 uses a similar trick to distinguish left from right. Tracks 3 through 6 can refer to Track 1 to determine which direction is left and which is right, since they only need to have non-reflection invariant rules when Track 1 has an arrow state nearby.

TILING with rotation symmetry: To show these cases are computable, use the algorithm for one-dimensional TILING. Once we find a path in one direction, we can use the symmetry to extend it to a tiling of the complete grid.

WEIGHTED TILING with rotation symmetry and periodic BC: In this case, the same solution works as for unweighted TILING: find an optimal path for one direction and reflect it to give an optimal tiling for the whole grid.

WEIGHTED TILING with rotation symmetry and open BC: We find the cheapest adjacent pair of tiles and use them to tile the whole grid with a checkerboard pattern.

WEIGHTED TILING with rotation symmetry and four-corners BC: We fill most of the grid using a checkerboard of the cheapest pair of tiles. However, we may need to put a different pattern in the corners in order to satisfy the boundary conditions. We show that it is sufficient to use corner inserts of constant size, so we can just consider all possible corner inserts to find if a tiling is possible.

6. Conclusion

We have shown that a class of classical tiling problems and 1-dimensional quantum Hamiltonian problems can be proven hard, even when the rules are translationally-invariant and the only input is the size of the problem. While this result was motivated by the desire to see if it could be hard to find the ground state in some physically interesting system, it is true that the tiling problem and Hamiltonian problem for which we prove hardness are not themselves particularly natural. Still, given that very simple cellular automata can be universal, it seems quite possible that even some very simple tiling and Hamiltonian problems are complete for NEXP and $\mathsf{QMA}_{\mathsf{EXP}}$ respectively.

Another interesting avenue to pursue would be to apply a similar idea to other problems. For instance, the game of go produces a PSPACE-complete problem [19]. However, the computational problem GO is defined by asking whether black can force a win given a particular board configuration as input. However, there is no guarantee that these board configurations would appear in a regular game of go, which starts from a blank board. A more natural problem arising from GO is to ask who wins with optimal play when starting from an empty $N \times N$ board. As with our tiling and Hamiltonian problems, the only thing that varies is the size; the rules and initial configuration are fixed. Thus, we would wish to show that this variant of GO is $\mathsf{EXPSPACE}$-complete. Our techniques will not solve this problem, but at least our result points the way to ask the right question.

Acknowledgements

We would like to thank John Preskill for pointing out the connection to N-REPRESENTABILITY, and Cris Moore and Igor Pak for information about classical tiling. D.G. was supported by Perimeter Institute for Theoretical Physics, CIFAR, and NSERC. Research at Perimeter Institute is supported by the Government of Canada through Industry Canada and by the Province of Ontario through the Ministry of Research & Innovation. S.I. was supported in part by NSF Grants CCR-0514082 and CCF-0916181. Part of this work was done while both authors were visiting the Institute for Quantum Information at Caltech.

References

[1] D. Aharonov, D. Gottesman, S. Irani, and J. Kempe, "The power of quantum systems on a line," in *FOCS*, 2007, pp. 373–383, arXiv:0705.4077v2 [quant-ph].

[2] S. Ben-David, B. Chor, and O. Goldreich, "On the theory of average case complexity," in *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1989, pp. 204–216.

[3] R. Berge, "The undecidability of the domino problem," *Memoirs of the Am. Math. Soc.*, vol. 66, pp. 1–72, 1966.

[4] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM Journal on Computing*, vol. 26, pp. 11–20, 1997.

[5] P. V. E. Boas, "The convenience of tilings," in *In Complexity, Logic, and Recursion Theory*. Marcel Dekker Inc, 1997, pp. 331–363.

[6] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer," *Royal Society of London Proceedings Series A*, vol. 400, pp. 97–117, Jul. 1985.

[7] M. Fürer, "The computational complexity of the unconstrained limited domino problem (with implications for logical decision problems)," in *Logic and Machines*, 1983, pp. 312–319.

[8] H. Galperin and A. Wigderson, "Succinct representations of graphs," *Inf. Control*, vol. 56, no. 3, pp. 183–198, 1983.

[9] D. Gottesman and S. Irani, "The quantum and classical complexity of translationally invariant tiling and hamiltonian problems," 2009. [Online]. Available: http://arXiv.org/abs/0905.2419

[10] S. Irani, "Ground state entanglement in one dimensional translationally invariant quantum systems," 2009, arXiv:0901.1107v1 [quant-ph].

[11] D. Janzing, P. Wocjan, and S. Zhang, "A single-shot measurement of the energy of product states in a translation invariant spin chain can replace any quantum computation," 2007, arXiv:0710.1615v2 [quant-ph].

[12] A. Kay, "The computational power of symmetric hamiltonians," *Phys. Rev. A*, vol. 78, p. 012346, 2008, arXiv:0801.3228v3 [quant-ph].

[13] A. Kitaev, A. Shen, and M. Vyalyi, *Classical and Quantum Computation*. Providence, RI: AMS, 2002.

[14] L. A. Levin, "Average case complete problems," *SIAM J. Comput.*, vol. 15, no. 1, pp. 285–286, 1986.

[15] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of Computation (2nd Edition)*, 2nd ed. Prentice Hall, August 1997.

[16] Y.-K. Liu, M. Christandl, and F. Verstraete, "N-representability is QMA-complete," *Phys. Rev. Lett.*, vol. 98, p. 110503, 2007, arXiv:quant-ph/0609125.

[17] D. Nagaj and P. Wocjan, "Hamiltonian quantum cellular automata in 1d," 2008, arXiv:0802.0886v4 [quant-ph].

[18] R. Oliveira and B. Terhal, "The complexity of quantum spin systems on a two-dimensional square lattice," 2005, quant-ph/0504050.

[19] C. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1995.

[20] C. H. Papadimitriou and M. Yannakakis, "A note on succinct representations of graphs," *Inf. Control*, vol. 71, no. 3, pp. 181–185, 1986.

[21] H. Wang, "Proving theorems by pattern recognition i," *Commun. ACM*, vol. 3, no. 4, pp. 220–234, 1960.