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

Daniel Gottesman       Sandy Irani

May 14, 2009

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 QMAEXP-complete. Thus, an algorithm for these problems which runs in time polynomial in $N$ (exponential in the input size) would imply that EXP = NEXP or BQEXP = QMAEXP, 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 [Lev86, BDCG89], 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 [AGIK07], 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 [AGIK07] 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 the 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 \( \text{NEXP} \)-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 \( \text{NEXP} = \text{EXP} \). While it is possible that \( \text{P} \neq \text{NP} \) and yet \( \text{NEXP} = \text{EXP} \), this seems unlikely to be the case.

This version of tiling is equivalent to the more common Wang Tiles [Wan60] 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 \( \text{NEXP} \)-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 [LP97][Boa97]. 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.

The basic idea of our construction is to reduce from an instance \( x \) of some language in \( \text{NEXP} \) 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. One of the postulates of quantum mechanics states that 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 are there-
fore interested in extending this even further and examining particles that are located in a geometrically $r$-dimensional space where only particles within a fixed distance can interact. A particularly natural model to consider, then, is 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 \cite{KSV02}. 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 $a - b \geq 1/\text{poly}(N)$, 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 \cite{OT05, AGIK07}.

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, \cite{NW08} gives a 20-state translation-invariant modification of the construction from \cite{AGIK07} (improving on a 56-state construction by \cite{JWZ07}) 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 \cite{Kay08} 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 \cite{Ira09}. 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 QMA$^\text{EXP}$, 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 section 5.3 for the algorithm). That is, it can be solved in a time $\text{polylog}(N)$, whereas it appears the quantum problem takes time $\text{exp}(N)$, even on a quantum computer (unless QMA$^\text{EXP} = \text{BQEXP}$, where BQEXP is like BQP, but with exponential circuits). Translational invariance does seem to simplify the 1-dimensional classical case, reducing $\text{poly}(N)$ time to $\text{polylog}(N)$ time, but it doesn’t help very much in the
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 $\langle t_i t_j \rangle \langle t_j t_i \rangle$ 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.

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, so we instead 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 are a critical component. We have chosen to fix the tiles used at the 4 corners of the square, but there are a number of other possible ways to set the boundary conditions, including 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. We show this case is also hard, but with a more complicated reduction than in our standard TILING problem. 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.

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 $(t_i, t_j) \in V$ 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. We do not
Determine whether the ground state energy of the system is at most that for each instance.

Machine can be of size $2^k$ on an input of length $n$.

Table 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.

Some variants of TILING we consider are easy but in a strange non-constructive sense in that there exists $N_0 \in \mathbb{Z}^+ \cup \{\infty\}$ such that if $N < N_0$, there exists a valid tiling, and if $N \geq N_0$, then there does not exist a tiling (or sometimes the other way around). However, $N_0$ is uncomputable as a function of $(T, H, V)$. These cases are denoted as “P, uncomputable” in Table 1 (with a question mark if 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 QMA$^{\text{EXP}}$. 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 $2^{kn}$ on an input of length $n$.

**Definition 2.3** A language $L$ is in QMA$^{\text{EXP}}$ iff there exists an $k$ and a Quantum Turing Machine $M$ such that for each instance $x$ and any $|\psi\rangle$ on $O(2^{k|x|})$ qubits, on input $(x, |\psi\rangle)$, $M$ halts in $O(2^{k|x|})$ 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 2.4** $r$-DIM TIH (Translationally-Invariant Hamiltonian)

**Problem Parameter:** $r$ Hamiltonian terms $H_1, \ldots, H_r$ 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 QMA_{exp} = BQEXP.

**Theorem 2.5** 1-DIM TIH is QMA_{exp}-complete.

The theorem immediately implies that r-DIM TIH is QMA_{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 prove theorem 2.5 in section 4. If we instead define r-DIM TIH with periodic boundary conditions, we get the same result.

As is common in QMA-completeness results, the construction for Theorem 2.5 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 \( N \)-state particles, is it within \( \epsilon \) of a state \( \rho' \) such that there exists a translationally-invariant pure state \( |\psi\rangle \) for \( N \) particles arranged in a circle for which \( \rho' \) is the marginal state of two adjacent particles? \( \rho \) 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 \( \rho \) to try. For each one, if we can solve \( N \)-REPRESENTABILITY, we can determine if \( \rho \) 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 \( N \text{tr}(H_{1}\rho) \). Trying all possible \( \rho \), we can thus find the ground state energy of \( H \), up to some \( \epsilon \)-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 \( f : \mathbb{Z} \to \{0, 1\}^{*} \) 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 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 in section 5 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 NTIME(\( 2^{c_{1}n} \)). If \( L \) is in NTIME(\( 2^{c_{2}n} \)), the reduction consists of padding an input \( x \) so that its length is \( |x|^{k}/c_{1} \) \( \text{[Pap95]} \). 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 a designated tile type \( \mathbb{H} \). (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 \( \mathbb{H} \) there are four other kinds of boundary tiles: \( \mathbb{1}, \mathbb{W}, \mathbb{M}, \mathbb{E} \). We will call the rest of the tiles interior tiles. \( \mathbb{1} \) will mark the left side of the grid, \( \mathbb{W} \) the top of the grid, \( \mathbb{M} \) the bottom of the grid, and \( \mathbb{E} \) the top side of the grid. (See figure 1.)

Nothing can go to the left of a \( \mathbb{1} \) tile which means that the only place a \( \mathbb{1} \) tile could go is the left-most boundary, as desired. Similarly, nothing can above a \( \mathbb{W} \) tile, nothing can go below a \( \mathbb{M} \) tile, and nothing can go to the right of a \( \mathbb{E} \) tile, which means \( \mathbb{W} \) tiles can only go in the top row, \( \mathbb{M} \) tiles can only go in the bottom row, and \( \mathbb{E} \) tiles can only go in the right-most column. No interior tile can border a \( \mathbb{H} \) tile in any direction. Furthermore a \( \mathbb{H} \) cannot border on itself in any direction. This means that the only possible locations for a \( \mathbb{H} \) are the four corners since those are the only places which can be surrounded by \( \mathbb{1}, \mathbb{W}, \mathbb{M}, \mathbb{E} \) tiles. Since the boundary conditions state that \( \mathbb{H} \) tiles must go in the corners, those are exactly the locations that will hold \( \mathbb{H} \) tiles. The only tiles that can go above or below the \( \mathbb{H} \) tiles are \( \mathbb{1} \) and \( \mathbb{W} \) tiles. The only tiles that can go either to the left or right of the \( \mathbb{H} \) tiles are \( \mathbb{M} \) and \( \mathbb{E} \) tiles. We will add the constraint that the only tiles that can go above or below \( \mathbb{1} \) tiles are \( \mathbb{W} \) tiles or \( \mathbb{M} \) tiles. Thus the entire west boundary, except for the corners, will be \( \mathbb{W} \) tiles. Similarly, we add constraints that \( \mathbb{M} \) tiles must have \( \mathbb{H} \) or \( \mathbb{1} \) tiles to their right and left and any \( \mathbb{E} \) tile must have a \( \mathbb{H} \) or \( \mathbb{M} \) tile to its right and left. This means that the entire south border except for the corners is tiled with \( \mathbb{M} \) tiles and the entire north border except for the corners will be \( \mathbb{W} \) 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 \( \mathbb{H} \) 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 \cite{Ber66}, 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 \( (\Sigma, Q, \delta) \), 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 \( \Sigma \) (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 pairings 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:

| \(c, q'\) | \(b, q', L\) | \(d\) |
| \(-\) | \([a, q]\) | \([d, q, L]\) |

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 \((a, q) \rightarrow (b, q', L)\), although there may have been other non-deterministic choices. The \([c, q']\) tile to the left of \([b, q', L]\) 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 \(\texttt{□}\) tile are \([q_0, \#]\) or \([\#]\) tiles. We forbid having a \([\#]\) tile to the right of a \(\texttt{□}\) 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 \(\texttt{□}\) tiles. A layer 2 tile that goes above a \(\texttt{□}\) must be \([a]\) or \([a, q_0]\) for some \(a \in \Sigma\). Furthermore, in the space above an \(\texttt{□}\) 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 \(\texttt{□}\) 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 \(\texttt{□}\) 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 \(\Sigma_M\) for every \(a \in \Sigma_{M_{BC}}\).

Finally, the only variety 2 tiles on layer 2 which we allow below a \(\texttt{□}\) 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 on input \(x\) 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 [BV97] 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 \(f : \mathbb{Z} \rightarrow \{0, 1\}^*\)
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 $x$.

We can reduce any language in QMA$_{\text{EXP}}$ 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 $\epsilon$ for $\epsilon = 1/poly(N)$ [KSV02]. 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 2.5 will then take input string $x$ and output an integer $N$ such that $f_{BC}(N - 3) = x$. 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 $\epsilon$, then the ground state energy of the system is at least $p(N) + 1/q(N)$.

Quantum Turing machines were first introduced in [Deu85] and further developed in [BV97]. 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 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 on input $0^n$, we will produce a Hamiltonian term $H$ which acts on a pair of finite-dimensional particles. We can use standard boosting techniques to assume that the probability of acceptance or rejection is at least $1 - \epsilon$ or at most $\epsilon$ for $\epsilon = 1/poly(N)$ [KSV02]. 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 2.5 will then take input string $x$ and output an integer $N$ such that $f_{BC}(N - 3) = x$. 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 $\epsilon$, then the ground state energy of the system is at least $p(N) + 1/q(N)$.

Quantum Turing machines were first introduced in [Deu85] and further developed in [BV97]. 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 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 $\langle$ and $\rangle$ 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 $\langle$, the right-most particle is in state $\rangle$, and no other particle in the chain is in state $\langle$ or $\rangle$. $S_{br}$ is the space spanned by all bracketed states.

We will restrict our attention for now to $S_{br}$ and add a term later that gives an energy penalty to any state outside $S_{br}$. 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. The figure below gives a picture of the start state for the system. Each element represents the state of an individual particle.

| $\langle$ | $\bullet$ | $\cdot \cdot \cdot$ | Track 1: Clock second hand | $\cdot \cdot \cdot$ | $\bullet$ | $\rangle$ |
| $\bullet$ | $\cdot \cdot \cdot$ | Track 2: Clock minute hand | $\cdot \cdot \cdot$ | $\bullet$ | $\cdot \cdot \cdot$ |
| $\# \ # \cdot \cdot \cdot$ | Track 3: Turing machine work tape | $\cdot \cdot \cdot$ | $\# \ # \cdot \cdot \cdot$ |
| $\cdot \cdot \cdot$ | Track 4: Tape head and state for TM $M_{BC}$ | $\cdot \cdot \cdot$ | $\cdot \cdot \cdot$ |
| $\cdot \cdot \cdot$ | Track 5: Tape head and state for TM $V$ | $\cdot \cdot \cdot$ | $\cdot \cdot \cdot$ |
| $0/1 \ # \ # \cdot \cdot \cdot$ | Track 6: Quantum witness for $V$ | $\cdot \cdot \cdot$ | $0/1 \ # \ # \cdot \cdot \cdot$ |
As is typical in hardness results for finding ground state energies, the Hamiltonian applied to each pair will consist of a sum of terms of which there are two types. 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 $N - 3$ steps, where $N$ is the length of the chain.

We will make use of the following simple lemma throughout the construction in limiting the set of standard basis states in the support of the ground state.

**Lemma 4.2** For any regular expression over the set of particle states in which each state appears at most once, we can use illegal pairs to ensure that any legal standard basis state for the system is a substring of a string in the regular set.

**Proof:** The alphabet for the regular expression is the set of particle states. Since each character appears once in the regular expression, the set of characters $b$ which can follow a particular character $a$ is well defined and does not depend on where the character appears in the string. Therefore, we can add an illegal pair $ab$ if $b$ is not one of the characters which can follow $a$. Any substring of the regular expression has no illegal pairs.

We give now a brief outline of the construction and provide the full details in the subsections that follow. Illegal pairs are used to enforce that the state of Track 1 is always of the form $\otimes \otimes^* (11 \oplus 00) \ominus^* \ominus$. (The $+$ denotes the regular expression OR and not a quantum superposition.) 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 $\otimes^* \otimes^* (00 \oplus 11) \ominus$. The second phase is the Computation Phase and consists of all $N - 2$ of the states of the form $\otimes^* \otimes^* (11 \oplus 00) \ominus$. The $\otimes^* \otimes^* (00 \oplus 11) \ominus$ states are ordered according to the number of particles in state $1$ and the $\otimes^* \otimes^* (11 \oplus 00) \ominus$ states are ordered according to the number of particles in state $2$. The state immediately after $\otimes^* \otimes^* (00 \oplus 11) \ominus$ in the ordering is $\otimes^* \otimes^* (11 \oplus 00) \ominus$. The target ground state for the clock is the uniform superposition of all the clock states, entangled appropriately with states of the other 4 tracks. 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: $\overline{1}$ and $\overline{2}$ will be used during the initial minute of the clock when it is in state $\otimes^* \otimes^* (11 \oplus 00) \ominus$ and will be used to check initial conditions on the other tracks. $\overline{1}$ and $\overline{2}$ will be used during the counting phase and $\overline{1}$ and $\overline{2}$ will be used during the computation phase. $\overline{1}$ and $\overline{2}$ will be used to trigger different actions on the other tapes. Every time the $\overline{1}$ 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. The \( \ominus \) symbol is what causes the Turing machine \( V \) to execute a step. 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 \( \ominus \) or \( \oplus \) 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.

### 4.1 The Clock Tracks

There will be four types of states in Track 1: \( \ominus \), \( \ominus \), \( \ominus \), and \( \ominus \). (However, the \( \ominus \) and \( \ominus \) states each come in multiple varieties — see below for the details.) We will have illegal pairs that will enforce that any standard basis state with no illegal pairs must be a substring of a string from the regular expression:

\[
\ominus \ominus^* (\ominus + \ominus) \ominus^\ominus.
\]

Furthermore, any bracketed standard basis state must be exactly a string corresponding to this regular expression. The transition rules then move the arrows in the direction in which they are pointing:

1. \( \ominus \ominus \rightarrow \ominus \ominus \), \( \ominus \ominus \rightarrow \ominus \ominus \): in the forward direction, arrows move in the direction they are pointing.
   In the reverse direction, they move in the opposite direction.

2. \( \ominus \ominus \rightarrow \ominus \ominus \), \( \ominus \ominus \rightarrow \ominus \ominus \): arrows change direction when they hit an endpoint.

Note that the \( \ominus \) symbol always interacts with the particle on its right in the forward direction and the particle on its left in the reverse direction. Similarly, the \( \ominus \) symbol interacts with the particle on its left in the forward direction and the particle on the right in the reverse direction. Therefore, we know that every configuration for Track 1 has exactly one transition rule that applies in the forward direction and one that applies in the reverse direction. The arrow symbol on Track 1 shuttles back and forth between the left end and the right end as shown below 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.

There are five states for Track 2: \( \ominus \), \( \ominus \), \( \ominus \) and \( \ominus \) and \( \ominus \). We will have illegal pairs that will enforce that any legal configuration on Track 2 must have the form \( \ominus \ominus^* (\ominus \ominus^* + \ominus \ominus^*) \ominus \). We will impose an ordering on the set of all possible such states and select transitions so that in one Track 1 iteration, the state on Track 2 will advance from one configuration to the next in the ordering. Each state for Track 2 corresponds to a minute in our clock and the states are partitioned into two phases. The first phase is called the **Counting Phase** and consists of all of the states of the form \( \ominus \ominus^* (\ominus \ominus^* + \ominus \ominus^*) \ominus \). The second phase is the **Computation Phase** and consists of all of the states of the form \( \ominus \ominus^* \ominus \ominus^* \ominus \). The \( \ominus \ominus^* \ominus \ominus^* \ominus \ominus^* \ominus \ominus^* \ominus \ominus^* \ominus \) states are ordered according to the number of particles in state \( \ominus \) and the \( \ominus \ominus^* \ominus \ominus^* \ominus \ominus^* \ominus \ominus^* \ominus \ominus^* \ominus \) states are ordered according to the number of particles in state \( \ominus \). The state immediately after \( \ominus \ominus^* \ominus \ominus^* \ominus \) in the ordering is \( \ominus \ominus^* \ominus \ominus^* \ominus \ominus^* \ominus \ominus^* \ominus \). The ordering of the states for Track 2 is shown below for a six particle system.
The arrow particle on Track 1 will cause a transition from one Track 2 state to the next. Each of the arrow states for Track 1 will come in three varieties: \( \rightarrow 0 \) and \( \rightarrow 0 \) will be used during the initial minute of the clock when it is in state \( \longleftarrow 0 \) and will be used to check initial conditions on the other tracks. \( \rightarrow 0 \) and \( \rightarrow 0 \) will be used during the counting phase and \( \rightarrow 0 \) and \( \rightarrow 0 \) will be used during the computation phase. We need to describe how these different Track 1 symbols trigger transitions on Track 2. The transitions for Track 1 will remain as described in that the transition \( \rightarrow 0 \rightarrow 0 \) will always cause a transition from one of the \( \rightarrow 0 \) symbols to one of the other \( \rightarrow 0 \) symbols, but we need to specify which \( \rightarrow 0 \) symbols are used in order to ensure that forward and backwards transitions remain unique for each standard basis state.

We will use ordered pairs to denote a combined Track 1 and Track 2 state for a particle as in \([\circ, 1]\). The states for the left-most and right-most particles are not divided into tracks and are simply \( \langle \rangle \) or \( \rangle \). As space permits we will denote the states for the different tracks vertically aligned. Examples for neighboring particle states are given below:

\[
\begin{array}{c}
\langle \circ \rangle \\
\rangle \\
\rangle
\end{array}
\]

We will use the symbol \( \bullet \) to denote a variable state. Thus \([\circ, \bullet]\) is used to denote any state in which the Track 1 state is \( \circ \).

**Clock States for the Initialization:** When Track 1 has a \( \circ \) or \( \circ \) symbol, we want to ensure that Track 2 is in state \( \langle 0 \rangle \langle 0 \rangle \langle \rangle \). Therefore, we disallow \([\circ, \bullet]\) and \([\circ, \bullet]\) for any \( \bullet \) that is not \( 0 \) or \( \circ \). We have the usual transitions that advance the Track 1 arrow:

\[
\begin{array}{c}
\circ \\
\bullet \\
\circ
\end{array}
\]

These happen regardless of the values on the other tracks and do not change the values on the other Tracks. The state \( \langle \bullet, 0 \rangle \) does not have a transition in the reverse direction since this only occurs in the initial state. Finally, we have the transition

\[
\begin{array}{c}
\langle \circ \rangle \\
\langle 0 \rangle
\end{array}
\]

The presence of the \( \circ \) on Track 2 specifies that \( \circ \) should transition to \( \circ \) in the reverse direction instead of \( \circ \). The initial iteration of the second hand is demonstrated on a chain of length six in figure 2.

**Clock States for the Counting Phase:** During the counting phase, Track 2 will always be in some state of the form \( \langle 1 \rangle \langle 0 \rangle \langle 0 \rangle \), so we will forbid the states \([\circ, \bullet]\) and \([\circ, \bullet]\) when \( \bullet \) is \( 2 \) or \( \circ \). The right-moving transitions will remain unchanged: \( \circ \circ \rightarrow \circ \circ \). These occur regardless of the contents of Track 2 and do not effect any change on Track 2. The change in direction at the right end will depend on the contents of Track 2:

\[
\begin{array}{c}
\circ \\
\circ
\end{array}
\]

The latter transition triggers the transition to the computation phase. In the left-moving direction, the arrow will sweep past pairs of \( \circ \) particles and pairs of \( \circ \) particles:

\[
\begin{array}{c}
\circ \\
\circ
\end{array}
\]

12
When the arrow on Track 1 sweeps left and meets the ⑥ particle on Track 2, it triggers an advance of the minute hand. This does not change the transition on Track 1:

\[
\begin{array}{c}
\text{⑥} \\
\text{00}
\end{array} \rightarrow \begin{array}{c}
\text{⑦} \\
\text{00}
\end{array}
\]

Note that the ⑦ never coincides with the ⑥, so we will disallow the state \([⑦, ⑥]\). Finally, we have that ⑦ must turn at the left end:

\[
\begin{array}{c}
\text{⑦} \\
\text{11}
\end{array} \rightarrow \begin{array}{c}
\text{⑦} \\
\text{11}
\end{array}
\]

We illustrate in figure 3 the first iteration of the counting phase and then the last iteration in the counting phase. The very last transition illustrates the transition to the computation phase.

**Clock States for the Computation Phase:** During this time, Track 2 will always be in some state of the form \(\text{②}\text{②}\text{②}\), so we will forbid the states \([⑦, ①]\) and \([⑦, ①]\) when ① is 0 or ②. Since ⑦ should never be in the same configuration as ⑥, we will also disallow the pair \([⑦, ②][①, ②]\). The left-moving transitions will remain unchanged: ⑦① → ⑦①. These occur regardless of the contents of Track 2 and do not effect any change on Track 2. The change in direction at the left end will happen as long as there is a ① left in Track 2:

\[
\begin{array}{c}
\text{⑦} \\
\text{11}
\end{array} \rightarrow \begin{array}{c}
\text{⑦} \\
\text{11}
\end{array}
\]

When the state on Track 2 becomes \(\text{②②②}\) and the left-moving arrow on Track 1 reaches the left end of the chain, we want the clock to stop since this is the very last state. Thus, there is no forward transition out of \(\text{⑦①}\). In the right-moving direction, the arrow will sweep past pairs of ① particles and pairs of ② particles:

\[
\begin{array}{c}
①① \\
11
\end{array} \rightarrow \begin{array}{c}
①① \\
11
\end{array} \quad \begin{array}{c}
①② \\
22
\end{array} \rightarrow \begin{array}{c}
①② \\
22
\end{array}
\]

When the arrow on Track 1 sweeps right and meets the ① particle on Track 2, it triggers an advance of the minute hand:

\[
\begin{array}{c}
①① \\
11
\end{array} \rightarrow \begin{array}{c}
①⑦ \\
12
\end{array}
\]
First iteration of the counting phase | Last iteration of the counting phase

| 0 0 0 0 | 0 1 0 0 |
| 0 0 0 0 | 1 1 1 0 |
| 0 0 0 0 | 1 1 1 0 |
| 0 0 0 0 | 1 1 1 0 |
| 0 0 0 0 | 1 1 1 0 |
| 0 0 0 0 | 1 1 1 0 |
| 1 0 0 0 | 1 1 1 0 |
| 1 0 0 0 | 1 1 1 0 |

Figure 3: The first iteration and last iterations of the counting phase. The very last transition illustrates the transition to the computation phase.
Note that since the $\bar{2}$ never coincides with the $\bar{1}$, we can make the state $[\bar{2}, \bar{1}]$ illegal. Finally, we have the turning at the right end:

$$\begin{array}{c}
\text{First iteration of the Computation Phase} \\
\begin{array}{c}
\text{Last iteration of the Computation Phase}
\end{array}
\end{array}$$

We illustrate in figure 4 the first iteration in the computation phase and then the last iteration. The very last state shown is the final state for the clock.

**Figure 4:** The first and last iterations of the computation phase. The very last state shown is the final state for the clock.

Before describing the states and transitions for the other tracks, we will pause to consider the Hamiltonian created so far. A single two-particle term will be the sum of the terms from the transition rules and the illegal pairs described so far. Suppose that a state of a particle is described only by its state on Tracks 1 and 2. Let $H_N$ be the system resulting from applying this term to every neighboring pair in a chain of $N$ particles. What is the dimension and structure of the ground state of $H_N$ restricted to $S_{br}$?

**Definition 4.3** We will say that a standard basis state is well-formed if the Track 1 state corresponds to a string in the regular language $\langle \circ \circ ^* (\bar{2} + \bar{0}) \circ ^* \rangle$ and if the Track 2 state corresponds to a string in the regular language $\langle \circ \circ ^* (\bar{0} \circ ^* + \circ \bar{2}^*) \rangle$. 
Note that if a standard basis state is well-formed it must be in $S_{br}$.

**Lemma 4.4** Consider a well-formed standard basis state. There is at most one transition rule that applies to the state in the forward direction and at most one that applies to the state in the reverse direction. Furthermore, the set of well-formed states is closed under the transition rules.

**Proof:** The rules are summarized in table 2. If only the Track 1 states are specified then the rule holds for any states on Track 2 and does not alter the Track 2 states. No pair appears twice on the left-hand side of a transition rule and no pair appears twice in the right-hand side of a transition rule. It can be verified that the application of a transition rule maintains the condition of being well-formed.

Consider the set of states that correspond to the valid clock states beginning with $\hat{\Theta} \Theta \odot$ on Track 1 and $\leq \Theta \odot$ on Track 2 and ending with $\hat{\Theta} \Theta \odot$ on Track 1 and $\leq \Theta \odot$ on Track 2. There are exactly $4(N - 2)^2$ such states in this sequence. Let $|\phi_{cl}\rangle$ be the uniform superposition of these states.

![Table 2: The transition rules for Tracks 1 and 2.](image)
Lemma 4.5  Consider a well-formed standard basis state $s$ that is not in the support of $|\phi_{\text{cl}}\rangle$. Then for some $i \leq 2N$, $s$ will reach a state with an illegal pair after $i$ applications of the transition rules in either the forward or reverse direction.

Proof: We will argue that for each well-formed standard basis state, it is either in the support of $|\phi_{\text{cl}}\rangle$, has an illegal pair or is within $2N$ transitions of a state which contains an illegal pair. In some cases, we have added penalties for a particular particle state (e.g., $[\overline{7}, 2]$). This can easily be handled with illegal pairs by making any pair illegal which contains that particular state.

Every Track 1 configuration of the form $\langle \overline{\overline{0}} \rangle (\overline{\overline{\overline{7}}} + \overline{\overline{\overline{7}}}) \overline{\overline{\overline{0}}} \overline{\overline{\overline{0}}} \overline{\overline{\overline{0}}} \rangle$ occurs in $|\phi_{\text{cl}}\rangle$. Moreover, for every standard basis state in the support of $|\phi_{\text{cl}}\rangle$, if the control state is $\overline{7}$ or $\overline{7}$, then Track 2 is in state $\langle \overline{6} \rangle \overline{\overline{0}} \overline{\overline{\overline{0}}} \rangle$. Now consider a state in which Track 2 has a $\overline{1}$, $\overline{1}$ or $\overline{2}$ particle and the control particle on Track 1 is $\overline{7}$ or $\overline{7}$. The control particle will transition (in either the forward or reverse direction) towards the $\overline{1}$, $\overline{1}$ or $\overline{2}$ particle on Track 2 and eventually they will coincide. This happens in fewer than $N$ moves and will result in an illegal state.

Now consider the standard basis states whose control particle is $\overline{7}$ or $\overline{7}$. Every possible combination of $\langle \overline{0} \rangle (\overline{\overline{7}} + \overline{\overline{7}}) \overline{\overline{0}} \overline{\overline{\overline{0}}} \overline{\overline{\overline{0}}} \overline{\overline{\overline{0}}} \rangle$ occurs in $|\phi_{\text{cl}}\rangle$ with every possible combination of $\langle \overline{1} \rangle \overline{\overline{0}} \overline{\overline{\overline{0}}} \overline{\overline{\overline{0}}} \overline{\overline{\overline{0}}} \rangle$ except those where the control particle is $\overline{7}$ and coincides with the $\overline{0}$. The state $[\overline{7}, \overline{0}]$ is an illegal particle state. So we now need to take care of the case where the control particle on Track 1 is $\overline{7}$ or $\overline{7}$ and Track 2 is a $\langle \overline{6} \rangle \overline{\overline{0}} \overline{\overline{\overline{0}}} \rangle$ state. Since it is illegal for a $\overline{7}$ or $\overline{7}$ to coincide with a $\overline{1}$ or $\overline{2}$ on Track 2, we can assume that the control particle on Track 1 is over a $\overline{1}$ particle on Track 2. In this case, it will transition in the forward direction in fewer than $2N$ steps (possibly turning at the left end of the chain) until it reaches the $\overline{1}$. This will result in a particle in state $[\overline{7}, \overline{1}]$ which is illegal.

Finally, consider the case where the control particle is $\overline{7}$ or $\overline{7}$. Every possible combination of the expression $\langle \overline{0} \rangle (\overline{\overline{7}} + \overline{\overline{7}}) \overline{\overline{0}} \overline{\overline{\overline{0}}} \overline{\overline{\overline{0}}} \overline{\overline{\overline{0}}} \overline{\overline{\overline{0}}} \rangle$ occurs in $|\phi_{\text{cl}}\rangle$ with every possible combination of $\langle \overline{1} \rangle \overline{\overline{1}} \overline{\overline{\overline{1}}} \overline{\overline{\overline{1}}} \overline{\overline{\overline{1}}} \overline{\overline{\overline{1}}} \overline{\overline{\overline{1}}} \rangle$ except those where the control particle is $\overline{7}$ and coincides with the $\overline{1}$. The state $[\overline{7}, \overline{1}]$ is illegal. So we now need to take care of the case where the control particle on Track 1 is $\overline{7}$ or $\overline{7}$ and Track 2 is a $\langle \overline{6} \rangle \overline{\overline{0}} \overline{\overline{\overline{0}}} \rangle$ state. Since it is illegal for a $\overline{7}$ or $\overline{7}$ to coincide with a $\overline{6}$ or $\overline{6}$ on Track 2, we can assume that the control particle on Track 1 is over a $\overline{1}$ particle on Track 2. In this case, it will transition in the forward direction in fewer than $2N$ steps (possibly turning at the left end of the chain) until it is just before the $\overline{6}$ on Track 2. This will result in a particle $[\overline{7}, \overline{1}] [\overline{0}, \overline{0}]$ which is illegal.

Lemma 4.6  $|\phi_{\text{cl}}\rangle$ is the unique ground state of $H_N|S_{br}\rangle$. All other eigenstates have an energy that is at least $\Omega(1/N^3)$.

Proof: The proof of this lemma follows from the standard techniques used for showing QMA-completeness of 1-dimensional Hamiltonians [AGIK07], so we only give a brief sketch here. The idea is that any standard basis state inside $S_{br}$ which is not well-formed will have energy at least one from the illegal pairs, so we can restrict our attention to well-formed states. Now we create a graph over well-formed standard basis states. There is an edge from state $a$ to state $b$ if $b$ can be reached from $a$ by the application of one transition rule in the forward direction. Lemma 4.4 implies that this graph is composed of disjoint directed paths. Call $H_{\text{trans}}$ the Hamiltonian resulting from the sum of all the terms from transition rules and $H_{\text{legal}}$ the Hamiltonian from illegal pairs. The subspace spanned by the states in a single path is closed under $H_{\text{trans}}$.
and $H_{\text{legal}}$. Furthermore, the matrix for $H_{\text{trans}}$ restricted to the states in a path looks like

$$
\begin{pmatrix}
\frac{1}{2} & -\frac{1}{2} & 0 & \cdots & 0 \\
-\frac{1}{2} & 1 & -\frac{1}{2} & 0 & \ddots \\
0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & \ddots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\
0 & \cdots & 0 & -\frac{1}{2} & 1 & -\frac{1}{2}
\end{pmatrix}
$$

where the dimension of the matrix is the length of the path. The matrix for $H_{\text{legal}}$ is diagonal in the standard basis. If the path contains no states with illegal pairs, then $H_{\text{legal}}$ is zero. The unique ground state has zero energy and is the uniform superposition of all standard basis states in the path. Moreover, the next highest energy state has energy at least $\frac{1}{L}$, where $L$ is the length of the path. In our case, Lemma 4.5 indicates there is exactly one path with no illegal states which corresponds to $|\phi_{cl}\rangle$. The length of this path is $O(N^2)$, so the next highest eigenvalue in this space is at least $\Omega(1/N^2)$.

Now consider a path with some illegal states and suppose that the ratio of illegal states in the path to the length of the path is $\frac{1}{s}$. Again, it is known [KSV02] that any state in the subspace spanned by the states in this path will have energy be at least $\frac{1}{s^3}$. By Lemma 4.5 we know that for any path which does not correspond to $|\phi_{cl}\rangle$ the ratio of illegal states to the total number of states at least $1/2N$, which means that any state in this subspace will have energy at least $\Omega(1/N^3)$.

The remaining subsections describe how the arrow on Track 1 interacts with the other four tracks. The only role of Track 2 is to record the time. It causes the control state on Track 1 to transition from the counting phase to the computation phase and finally to stop iterating at the end of the computation phase.

### 4.2 Initialization Phase

Track 3 will be used as the tape for both Turing machines. Therefore the set of track 3 states for a particle consists of the union of the two alphabets for the two Turing machines. We want to ensure that this track is initialized to all blank symbols, so we disallow any state in which Track 1 is in $\rightarrow 0$ and Track 3 has anything but $\circ$.

Track 4 will store the location and state for $M_{BC}$. Therefore, we will add illegal pairs to ensure that the Track 4 state for the system always has the form $\bigcirc \circ^* \circ \bigcirc^* \bigcirc$, where $\bigcirc$ is a state for $M_{BC}$. We assume that $M_{BC}$ has a designated start state $\otimes$, so we want to ensure that Track 4 starts out in the configuration $\bigcirc \otimes \bigcirc^* \bigcirc$. To do this, we make any state illegal in which the Track 1 state is $\rightarrow 0$ and the Track 4 state is anything except $\otimes$ or $\bigcirc$. Furthermore, for any pair in which the left particle is in state $\bigcirc$, and the Track 1 state is $\rightarrow 0$, the Track 4 state must be $\otimes$. For any pair in which the left particle is not in state $\bigcirc$, and the Track 1 state is $\rightarrow 0$, the Track 4 state must be $\bigcirc$. The initial conditions for Track 5 are similar, except that the starting state for $V$ is used. Since Track 6 holds the quantum witness for $V$, it can be any state on $n$ qubits. A state of a particle is now specified by a 6-tuple.

### 4.3 Counting Phase

Every time the $\rightarrow 1$ 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. Since the classical, reversible Turing machine $M_{BC}$ is a special case of a quantum Turing machine, we describe the details for the more general case. The $\rightarrow$ symbol is what causes the Turing machine $V$ to execute a step.
4.4 Computation Phase

We will examine a particular quantum rule and explain the desired behavior of our machine. Consider a pair \((q,a)\), where \(q \in Q\) and \(a \in \Sigma\). \(a\) will encompass the state on the work tape as well as the state on the witness tape. Since the \(\Uparrow\) particle triggers the execution of a step, we will consider particles as triplet states of the form \([\Uparrow, q, a]\), \([\bigcirc, q, a]\) and \([\bigotimes, q, a]\), for \(a \in \Sigma\) and \(q \in Q \cup \{\bigcirc\} \cup \{\bigotimes\}\). \(q\) specifies the Track 5 state, and \(a\) tells us the state of Track 3 and Track 6. It will be convenient to refer to \(q\) as a generic Track 5 state of a particle which could be a state from \(Q\) as well as \(\bigcirc\) or \(\bigotimes\). We will also be interested in the computation state of a particle which just consists of a pair \([q,a]\), where \(q \in Q \cup \{\bigcirc\} \cup \{\bigotimes\}\) and \(a \in \Sigma\).

If the TM is in state \(q\) and in a location with an \(a\) on the tape, the QTM defines a superposition of possible next moves. Let \(\delta(q,a,p,b,D)\) denote the amplitude that the next state will be a configuration in which the state is \(p\), the tape symbol is overwritten by \(b\) and the head moves in direction \(D\). We will use a fact established in [BV97] that the states of a quantum Turing machine \(Q\) can be partitioned into two sets \(Q_L\) and \(Q_R\) such that states in \(Q_L\) can be reached only by moves in which the head moves left and \(Q_R\) can be reached only by moves in which the head moves right. We will use \(q_L\) to designate a generic element of \(Q_L\) and \(q_R\) to designate a generic element of \(Q_R\).

We will need to execute the moves in which the head moves left separately from the moves in which the head moves right. In order to do this, we introduce a new state \(q_R'\) for every state \(q_R \in Q_R\). We will call this set \(Q'_R\). It will be forbidden to have the Turing machine head in a state from \(Q'_R\) without also having the \(\Uparrow\) state on Track 1 in the same location. It will also be forbidden to have the \(\Uparrow\) in the same location as \(q\) if \(q \in Q_L\). We will be interested in two particular subspaces defined on the computation state for a pair of neighboring particles \([q,a][p,b]\), where \(a,b \in \Sigma\) and \(q,p \in Q \cup \{\bigcirc\} \cup \{\bigotimes\}\). \(S_A\) is the space spanned by all such legal basis states such that \(q \not\in Q_L\), \(p \not\in Q'_R\). Note that if the Track 1 state for a pair of particles is \(|\Uparrow \bigcirc\rangle\), then the only possible computation states for the pair are in \(S_A\). We also define \(S_B\) to be the space spanned by all legal basis states of the form \([q,a][p,b]\) such that \(p \not\in Q_L\), \(q \not\in Q'_R\). Similarly, if the Track 1 state for a pair of particles is \(|\Uparrow \bigotimes\rangle\), then the only possible legal computation states for the pair are in \(S_B\).

We will describe a transformation \(T\) that maps \(S_A\) to \(S_B\). The extended map \(T \otimes |\Uparrow \bigcirc\rangle \langle \Uparrow \bigotimes|\) as it is applied to each pair of particles in turn will implement a step of the Turing machine. We will call this extended map a transition rule since it carries one state to another in the same way that the transition rules for the clock state did. The difference now is that the states will in general be quantum states.

The transformation on \(S_A\) works in two parts. At the moment that the Track 1 arrow moves from the position to the left of the head, we execute the move for that location. For every \((q_L,b,L)\), with amplitude \(\delta(q,a,q_L,b,L)\), we write a \(b\) in the old head location and move the head left into state \(q_L\). For every \((q_R,b,R)\), with amplitude \(\delta(q,a,q_R,b,R)\), we write a \(b\) in the old head location, transition to state \(q'_R\) and leave the head in the same location. We need to defer the action of moving the head right until we have access to the new location. In the next step of the clock, when the \(\Uparrow\) is aligned with the state \(q'_R\), we move it right and convert it to \(q_R\).

That is, we want a sequence of two transitions.

\[
\begin{bmatrix}
\Uparrow \\
\bigcirc \\
q \\
a
\end{bmatrix} \rightarrow \sum_{a,b,q_L,q} \delta(q,a,q_L,b,L) \begin{bmatrix}
\Uparrow \\
\bigcirc \\
q_L \\
b
\end{bmatrix} + \sum_{a,b,q_R,q} \delta(q,a,q_R,b,R) \begin{bmatrix}
\Uparrow \\
\bigotimes \\
q_R \\
b
\end{bmatrix}.
\]

After this step, the configuration is a superposition of states in which the step has been performed on the configurations with the head in the same location as the \(\Uparrow\) state, except that moving the head to the right has been deferred. If the TM state is primed and is aligned with the location of the \(\Uparrow\), that triggers the execution of the right-moving step.

\[
\begin{bmatrix}
\Uparrow \\
\bigotimes \\
q_R \\
\bigotimes
\end{bmatrix} \rightarrow \begin{bmatrix}
\bigotimes \\
\bigotimes \\
q_R \\
\bigotimes
\end{bmatrix}.
\]
Otherwise, the $\oplus$ state just sweeps to the right, leaving the other tracks unchanged:

\[
\begin{array}{c}
\begin{array}{c}
\text{\begin{tabular}{c}
\hline
\hline
\hline
\end{tabular}}
\end{array}
\end{array}
\rightarrow \begin{array}{c}
\begin{array}{c}
\text{\begin{tabular}{c}
\hline
\hline
\hline
\end{tabular}}
\end{array}
\end{array}
\rightarrow \begin{array}{c}
\begin{array}{c}
\text{\begin{tabular}{c}
\hline
\hline
\hline
\end{tabular}}
\end{array}
\end{array}
\rightarrow \begin{array}{c}
\begin{array}{c}
\text{\begin{tabular}{c}
\hline
\hline
\hline
\end{tabular}}
\end{array}
\end{array}
\end{array}
\]

Note that transformations (1), (2) and (3) define $T$ on every state in $S_A$. Furthermore, the image of $T|_{S_A}$ is in $S_B$. A critical fact is that after $T$ is applied to a pair, the computation state of that pair is in $S_B$, which implies that the next pair over is now in subspace $S_A$ and $T$ can be applied to this new pair. We need to now show that $T$ has the necessary properties to be expressed as a Type II Hamiltonian term with the $|cd\rangle$ final state a superposition. That is, we want to show that $T$ is a partial isometry (meaning it preserves inner products) when restricted to $S_A$. Since $S_A$ and $S_B$ have the same dimension, this means that $T^\dagger$ is well-defined and a partial isometry on $S_B$. We can extend $T$ to be a unitary map on the full Hilbert space. Then we select the Hamiltonian term to be:

$$I_{S_A} \otimes |\oplus\oplus\rangle\langle\oplus\oplus| + I_{S_B} \otimes |\oplus\oplus\rangle\langle\oplus\oplus| - T \otimes |\emptyset\emptyset\rangle\langle\emptyset\emptyset| - T^\dagger \otimes |\oplus\oplus\rangle\langle\oplus\oplus|.$$ 

Expressions (1), (2) and (3) each define $T$ on a different orthogonal subspace of $S_A$. $S_{A1}$ is the space of the form $\langle q, a | q, a \rangle$, $S_{A2}$ is the space of states of the form $|q'_{R}, \emptyset\rangle|\emptyset, \emptyset \rangle$ and $S_{A3}$ is the space spanned by states of the form $|q_{R}, \emptyset\rangle|\emptyset, \emptyset \rangle$, $|\emptyset, \emptyset\rangle|\emptyset, \emptyset \rangle$ and $|\emptyset, \emptyset\rangle|\emptyset, \emptyset \rangle$. We will argue that $T$ is a partial isometry when restricted to each of $S_{A1}$, $S_{A2}$ and $S_{A3}$. Furthermore the images of $T_{S_{A1}}$, $T_{S_{A2}}$ and $T_{S_{A3}}$ are mutually orthogonal. This implies that $T$ is a partial isometry on $S_A$.

$T$ is clearly a partial isometry on each of $S_{A2}$ and $S_{A3}$. We only need to examine the transformation given in (1). We know that the application of a move of the QTM is unitary. In particular, it is unitary when restricted to the space of all configurations with the head in a particular location. This means that the transformation (1) ($T$ restricted to states of the form $\langle q, a| q, a \rangle$) is a partial isometry.

Finally, we must compare $S_A$ in different locations. In particular, when the TM head moves right from location $i$, it can end up in the same location as when it moves left from location $i+2$. In our implementation, when each step is implemented, these steps are orthogonal because the Track 1 state differs. They are also orthogonal in the original TM rule, since after the head moves right, it will be in a state from $Q_R$, whereas after it moves left, it will be in a state from $Q_L$, which is orthogonal to $Q_R$. Thus, our implementation is globally performing the correct TM rule.

We have argued that the extended transformations (or transition rules) which implement the Turing Machine behave as we would like them to as long as the states to which they are applied obey certain conditions. The following definition formalizes those conditions.

**Definition 4.7** We say that a state is invalid if one of the following conditions holds:

1. The control particle on Track 1 is on site $i$ and is in state $\oplus$. Furthermore, for some particle other than $i$, its Track 4 state is $q'_R$ for some $q'_R \in Q'_R$.

2. The control particle on Track 1 is on site $i$ and is in state $\ominus$. Furthermore, the Track 4 state for particle $i$ is $q_L$ for some $q_L \in Q_L$.

3. The control particle on Track 1 is on site $i$ and is in state $\ominus$. Furthermore, for some particle other than $i$, its Track 5 state is $q'_R$ for some $q'_R \in Q'_R$.

4. The control particle on Track 1 is on site $i$ and is in state $\ominus$. Furthermore, the Track 5 state for particle $i$ is $q_L$ for some $q_L \in Q_L$.

We say that a standard basis state is valid if it is not invalid. Any invalid state has an energy penalty of at least one by the illegal pairs added in this subsection. The subspace of all valid states is closed under
the action of the transition rules. As before, we call a state well-formed if its clock state (Track 1 and 2) is well-formed. For any valid, well-formed state, the transition rules apply non-trivially to only one pair of particles. Furthermore, the transition rules are reversible and norm-preserving over the space spanned by valid, well-formed states.

4.5 Combining the Tracks

We have described a set of transition rules for Tracks 1 and 2 which advance a clock through $T = 4(N-2)^2$ states. We have also described a set of transition rules which implement two Turing Machines on tracks 3 through 6. In each set of transition rules, the control particle of the clock on Track 1 advances one space. Thus, when they are combined, each transition rule advances the clock and implements part of a step on one of the Turing Machines. Consider any state such that the states for Tracks 1 and 2 are clock states and the states for Tracks 3 through 6 are a quantum state within the subspace of valid states. If the clock state is not the final state, then the transition rules apply to exactly one pair of particles and take the state to a unique new state with the same norm in which the clock has advanced by one tick. Similarly, if the clock state is not the initial state, then the transition rules applied in the reverse direction apply to exactly one pair of particles and take the state to a new state with the same norm in which the clock has gone back by one tick.

We argued in Subsection 4.1 that a zero eigenstate of the Hamiltonian must be a superposition of the sequence of clock states. Now that we are considering the states of the other tracks as well, this is a higher-dimensional space. In order to define a basis for the zero eigenspace of the Hamiltonian defined so far, we specify a standard basis state for Tracks 3 through 6 for the initial clock state. For example, we specify a standard basis state for Track 6. Let $|t⟩$ be the state of the clock after $t$ transitions. Define $|φ_{t,X,Y}⟩$ denote the state of Tracks 3 through 6 after $t$ applications of the transition rules assuming that Tracks 3 through 5 start in state $X$ and Track 6 starts in state $Y$. Define $|φ_{X,Y}⟩ = ∑_{t=0}^{T−1} |t⟩|φ_{t,X,Y}⟩$. The $|φ_{X,Y}⟩$ satisfy all the constraints due to transition rules and illegal states for Tracks 1 and 2. We will use $I$ to denote the desired initial configuration for Tracks 3 through 5. That is, $O^*$ on Track 3 and $⊗O^*$ and $g_0O^*$ on Tracks 4 and 5. The energy for any $|φ_{X,Y}⟩$ where $X ≠ I$ will be at least $Ω(1/N^2)$. This is because if $X ≠ I$, at least one state in the initialization phase will have an energy penalty as the control state sweeps to the right.

4.6 Boundary Conditions

First we address the case of a finite chain and add a term that will enforce that the ground state is in $S_{br}$. In order to penalize states that are not bracketed, we weight $H$ by a factor of 3 and add in the term $I − 3|⊗⟩⟨⊗| − |⊗⟩⟨⊗|$ to every particle. This has the effect of adding energy $N - 2$ to every state in $S_{br}$. The ground space is still spanned by the $|φ_{I,Y}⟩$ and the spectral gap remains $Ω(1/N^3)$. Any standard basis state outside $S_{br}$ will have a cost of at least $N - 1$: If there are $a$ particles in state $⊗$ or $⊗$ in the middle of the chain, they will save $a$ from the $I − 3|⊗⟩⟨⊗| − |⊗⟩⟨⊗|$ term. However, there will be at least $[a/2]$ illegal pairs because the $⊗$ particles in the middle will each form an illegal pair with the particle to their left and the $⊗$ particles in the middle will each form an illegal pair with the particles to their right. We can lower bound the additional cost by counting the type (⊗ or $⊗$) which is more plentiful, yielding at least $[a/2]$ distinct illegal pairs. Thus, there will be a net additional energy cost of at least $a/2$ when $a$ is even, or $(a + 3)/2$ when $a$ is odd.

If we instead have periodic boundary conditions, the problem is still QMA$_{EXP}$-complete. We start by removing the penalty for the pair $⊗⊗$. The set of legal and well-formed states is exactly the same as it was for the finite chain except that we can now have more than one segment around the cycle. For example, we
could have the following state wrapped around a cycle:

\[
\begin{array}{ccc}
\leftarrow & \cdots & \leftrightarrow \\
\text{Segment 1} & \text{Segment 2} & \text{Segment 3}
\end{array}
\]

Note that the transition rules do not change the locations or numbers of $\leftarrow$ or $\rightarrow$ sites, so we can restrict our attention to subspaces in which the segments are fixed. If a standard basis state is well-formed then every occurrence of $\rightarrow$ has a $\leftarrow$ to its immediate right and every occurrence of $\leftarrow$ has a $\rightarrow$ to its immediate left. Thus, we can assume that a standard basis state in the support of a ground state can be divided into valid segments. Of course, it is possible that there are no segments in which case the state could simply be a single particle state repeated around the entire cycle. We know that the states within a segment must be ground states for a chain of that length. Otherwise, the energy of the state is at least $\Omega(1/l^3)$, where $l$ is the length of the segment. We need to add additional terms and states which give an energy penalty to states with no segments or more than one segment.

It is most straightforward to achieve this by using odd $N$ and adding an additional Track 0. There will be no transition rules that apply to Track 0, so the state of the Track 0 remains fixed even as transition rules apply to the state. We will add some extra terms which are diagonal in the standard basis for Track 0 and introduce energy penalties for certain pairs of particles. Fix a standard basis state for Track 0 and a set of segments and consider the space spanned by these states. The Hamiltonian is closed over this space, so we just need to examine the eigenstates restricted to each such subspace. Track 0 will have three possible states $A$, $B$, and $I$. Even the $\rightarrow$ or $\leftarrow$ particles have a Track 0 state, so the state of a particle is either in $\{A, B, I\} \times \{\leftarrow, \rightarrow\}$ or it is a 7-tuple denoting states for Tracks 0 through 6.

We add the terms $|A\rangle \langle A| \otimes |\leftarrow\rangle \langle \leftarrow|$ and $|B\rangle \langle B| \otimes |\rightarrow\rangle \langle \rightarrow|$, which gives an energy penalty of 1 for pairs $A\otimes I$ or $B\otimes I$ on Track 0. Since $N$ is odd, the state will have energy at least one unless there is at least one particle whose Track 0 state is $I$. Now we add an energy penalty of 1 for any particle whose Track 0 state is $I$. Note that the state will have energy less than one is for there to be at least one segment and to have a $I$ on Track 0 exactly at the locations where there is a $\leftarrow$ on the other tracks. Finally, we add an energy penalty of $1/2$ for any particle whose Track 0 state is $I$. Thus, the only way for a state to have energy less than one is for there to be exactly one segment and to have a $I$ on Track 0 exactly at the single location where there is a $\leftarrow$ on the other tracks.

Consider the set of all basis states with exactly one segment and whose Track 0 state has a $I$ co-located with the $\leftarrow$ site and alternating $A$’s and $B$’s elsewhere on Track 0. Any eigenstate outside this space has energy at least one. Note that there are $2N$ different low-energy ways to arrange the Track 0 states, since there are $N$ possible locations for the $I$ site and the string of alternating $A$’s and $B$’s can begin with $A$ or $B$. Each such choice gives rise to a subspace that is closed over $H$ and otherwise has identical behavior with respect to the rest of the terms, so we will make an arbitrary choice and examine the resulting subspace. Since there is one segment, the state $|\phi_{I,Y}\rangle$ from the previous subsection is well defined. This state is a ground state for $H$ and has energy 1/2. All other eigenstates within the subspace have energy at least $1/2 + \Omega(1/N^3)$.

### 4.7 Accepting States

Finally, we add a term for those $|\phi_{I,Y}\rangle$ that do not lead to an accepting computation for $V$. In the last configuration of the clock, the two right-most particles are in state $|\leftarrow\rangle \otimes |\rightarrow\rangle$. We assume that an accepting computation will end with the right particle in the left-most place in state $q_A$. We add a penalty for a pair whose Track 1 and 2 state is $|\leftarrow\rangle \otimes |\rightarrow\rangle$ such that the right particle in the pair does not have a Track 5 state of $q_A$. Thus, an accepting computation will have an energy penalty of at most $\epsilon/N^2$ for this term where $\epsilon$ can be made arbitrarily small. If no witness leads to an accepting computation, all states will have energy at least $(1 - \epsilon)/N^2$ from this term.
5 Variants of the Classical Tiling Problem

We now turn to studying variants of TILING. There are variety of different boundary conditions we could consider. We can also consider changing the absolute prohibitions on certain adjacent tiles to a soft condition by assigning different weights to the different adjacent pairs of tiles. We can add additional reflection or rotation symmetry to the tiling rules. We do not have definitive results on the complexity of all combinations of these variations, but we have been able to determine the complexity of most combinations, and in this section we present our analysis. Also, we prove that in one dimension, TILING and all the above variants are easy.

5.1 Boundary Conditions

Our choice of fixing the tiles in all four corners of the square is unusual. See, for instance, Papadimitriou [Pap95], who fixes just the tile in a single corner. If we had instead chosen that convention, the TILING problem would become easy, in an annoying non-constructive sense:

**Theorem 5.1** Define a variant of TILING where a designated tile $t_1$ is placed in the upper right corner of the grid, and the other corners are unconstrained. Then there exists $N_0 \in \mathbb{Z}^+ \cup \{\infty\}$ such that if $N < N_0$, there exists a valid tiling, and if $N \geq N_0$, then there does not exist a tiling. However, $N_0$ is uncomputable as a function of $(T, H, V)$.

This theorem follows immediately from the observation that with this boundary condition, 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. We know $N_0$ must be uncomputable because the question of whether there is an infinite tiling is uncomputable. Still, if we fix $(T, H, V)$, 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.

On the other hand, if we fix boundary conditions in two corners, that is already enough for hardness. Our proof (section 3) uses pre-determined tiles in two adjacent corners, but two opposite corners would suffice as well. Suppose they are the top right and bottom left corners, and $t_1$ is the tile type fixed for the upper right and lower left corners. Then we can introduce three additional tile types $t_2$, $t_3$, and $t_4$. We allow only $t_2$ to be to the right of $t_3$ and only $t_3$ or $t_4$ to be to the right of $t_2$. We also allow only $t_3$ to be below $t_2$ and only $t_3$ or $t_4$ to be below $t_2$. There is one more rule that says that $t_3$ cannot be above $t_2$ or to the left of $t_2$. There are other pairs, such as $t_3$, which we can either forbid or allow; they will not occur regardless. Naturally, the other tiles used in the construction will have allowed and forbidden configurations between themselves as before.

Thus, the bottom row will be all $t_2$ or $t_3$ (except for the left end), and the right column will be all $t_3$ or $t_4$ (except for the top). Then the bottom right corner can only be $t_4$, fixing the tile there as well. Indeed, $t_4$ can only be in the bottom right corner, so the bottom row is all $t_2$ except for the corners, and the right column is all $t_3$ except for the corners. See figure 5.

Another interesting case is when we have periodic boundary conditions. That is, we consider the top row to be adjacent to the bottom row, and the leftmost column is adjacent to the rightmost column. Essentially,
we are tiling a torus. This case is particularly interesting because it is truly translationally invariant, unlike our usual formulation, where the boundaries break the translational invariance.

**Theorem 5.2** Define PERIODIC TILING as a variant of TILING with periodic boundary conditions on an \( N \times N \) grid. PERIODIC TILING is NEXP-complete under an expected poly-time randomized reduction or a deterministic polyspace reduction.

Since we are dealing with the class NEXP, even an exponential-time reduction is meaningful. For instance, if there is an algorithm to solve PERIODIC TILING which takes time \( \text{poly}(N) \), then, as a consequence of theorem 5.2, \( \text{EXP} = \text{NEXP} \).

**Proof:** To prove this, we will show that, for appropriate \( N \), we can introduce an effective horizontal and vertical border at some point inside the square. The actual location of the borders will not be specified at all, but we will choose conditions so that there is exactly one horizontal border and one vertical border. Those borders will then act like the usual edges of the grid for the standard 4-corners boundary condition.

We will specialize to \( N \) which is an odd prime. We will add two additional layers of tiles over those for the usual TILING NEXP-completeness result (section 3). The new layer 1 has 7 different possible types of tile: \( \text{[tile1]} \), \( \text{[tile2]} \), \( \text{[tile3]} \), \( \text{[tile4]} \), \( \text{[tile5]} \), \( \text{[tile6]} \), and \( \text{[tile7]} \). The new layer 2 has 10 types of tile: \( \text{[tile8]} \), \( \text{[tile9]} \), \( \text{[tile10]} \), \( \text{[tile11]} \), \( \text{[tile12]} \), \( \text{[tile13]} \), \( \text{[tile14]} \), \( \text{[tile15]} \), and \( \text{[tile16]} \).

The \( \text{[tile17]} \) and \( \text{[tile18]} \) tiles will create the vertical and horizontal borders. \( \text{[tile19]} \) and \( \text{[tile20]} \) are used to make sure that there is at least one of each kind of border, and the \( \text{[tile21]} \) and \( \text{[tile22]} \) tiles will create a diagonal line within the rectangle defined by the borders. Layer 2 is used to mark the directions next to the border and make sure the diagonal line goes from the upper left corner of the rectangle to the bottom right corner, which ensures that the rectangle is a square. When \( N \) is prime, this means there can only be one horizontal and one vertical border.

**Layer 1:** \( \text{[tile23]} \) can only be adjacent horizontally to \( \text{[tile24]} \) or \( \text{[tile25]} \) and \( \text{[tile26]} \) can only be adjacent vertically to \( \text{[tile27]} \) or \( \text{[tile28]} \). \( \text{[tile29]} \) can only be adjacent vertically to \( \text{[tile30]} \) and \( \text{[tile31]} \) and can only be adjacent horizontally to \( \text{[tile32]} \). These rules imply that if we have a \( \text{[tile33]} \) anywhere in a column, that column must only contain \( \text{[tile34]} \) and \( \text{[tile35]} \) tiles, and if we have a \( \text{[tile36]} \) anywhere in a row, that row can only contain \( \text{[tile37]} \) and \( \text{[tile38]} \) tiles. Thus, layer 1 contains some number of vertical and horizontal lines composed of \( \text{[tile39]} \) and \( \text{[tile40]} \) tiles intersecting at \( \text{[tile41]} \) tiles. No two horizontal or vertical lines can be adjacent.

The next set of rules for layer 1 enforce that the space between the lines must be filled by a checkerboard of \( \text{[tile42]} \) or \( \text{[tile43]} \) alternating with \( \text{[tile44]} \) or \( \text{[tile45]} \) tiles. We ensure this by forbidding \( \text{[tile46]} \) and \( \text{[tile47]} \) from being adjacent to themselves or each other in any direction, and forbidding \( \text{[tile48]} \) and \( \text{[tile49]} \) from being adjacent to themselves or each other in any direction. Since \( N \) is odd, this ensures that it is not possible to tile the entire torus with the checkerboard pattern and there must be at least one horizontal and one vertical line.

Furthermore, if we ever have a \( \text{[tile50]} \) tile anywhere, we want to be forced to have a diagonal line of \( \text{[tile51]} \) tiles continuing to the upper left and lower right, with a diagonal line of \( \text{[tile52]} \) tiles next to it (above and to the right), with both lines ending at a vertical or horizontal line. We enforce this by requiring that a \( \text{[tile53]} \) tile must have \( \text{[tile54]} \) both below it and to its left. Above a \( \text{[tile55]} \) tile we can have only \( \text{[tile56]} \) or \( \text{[tile57]} \), and to the right of a \( \text{[tile58]} \) tile, we must have either \( \text{[tile59]} \) or \( \text{[tile60]} \).

**Layer 2:** We wish \( \text{[tile61]} \) to mark the top side of the rectangles delineated by the layer 1 \( \text{[tile62]} \) and \( \text{[tile63]} \) tiles. \( \text{[tile64]} \) will mark the right side, \( \text{[tile65]} \) the left side, and \( \text{[tile66]} \) the bottom side, with \( \text{[tile67]} \), \( \text{[tile68]} \), \( \text{[tile69]} \), and \( \text{[tile70]} \) marking the upper left, upper right, lower left, and lower right corners, respectively. \( \text{[tile71]} \) will be inside the rectangle and \( \text{[tile72]} \) will go over the layer 1 \( \text{[tile73]} \) and \( \text{[tile74]} \) tiles.

\( \text{[tile75]} \) can have \( \text{[tile76]} \) adjacent to it in either horizontal direction; it could also have \( \text{[tile77]} \) to the left and \( \text{[tile78]} \) to the right. A \( \text{[tile79]} \) tile can have \( \text{[tile80]} \) adjacent in either horizontal direction, and could have \( \text{[tile81]} \) to its left or \( \text{[tile82]} \) to its right. \( \text{[tile83]} \) can have \( \text{[tile84]} \) adjacent to it in either vertical direction, or could have \( \text{[tile85]} \) above or \( \text{[tile86]} \) below. \( \text{[tile87]} \) can have \( \text{[tile88]} \) adjacent to it in either vertical direction, or could have \( \text{[tile89]} \) above or \( \text{[tile90]} \) below. \( \text{[tile91]} \) must have a \( \text{[tile92]} \) to its right and a \( \text{[tile93]} \) below it. \( \text{[tile94]} \) must have a \( \text{[tile95]} \) to its left and a \( \text{[tile96]} \) below it. \( \text{[tile97]} \) must have a
above it and a \[ \square \] to the right of it. \[ \square \] must have a \[ \square \] to its left and a \[ \square \] above it. Thus, if we have a \[ \square \] anywhere, there must be a horizontal line of \[ \square \] that either goes all the way around the grid (due to the periodic boundary conditions) or ends at \[ \square \] on the left and \[ \square \] on the right. If it ends, then below the \[ \square \] tile is a vertical line of \[ \square \] tiles which end at a \[ \square \] tile, and below the \[ \square \] tile is a vertical line of \[ \square \] tiles which end at a \[ \square \] tile. The \[ \square \] and \[ \square \] tiles must be in the same row, and between them is a line of \[ \square \] tiles. Following this line of logic for the other tiles, we find that layer 2 must be a collection of rectangles, horizontal stripes, and vertical stripes.

Now we add rules that enforce that we have \[ \square \] on the outside of the rectangles and \[ \square \] on the inside. That is, if any of the border tiles has a solid color on one edge, it must be adjacent on that side to a matching solid color tile. \[ \square \] must have \[ \square \] above it and \[ \square \] below it, and it is the other way around for \[ \square \] (\[ \square \] below and \[ \square \] above). \[ \square \] must have \[ \square \] to the left and \[ \square \] to the right, and \[ \square \] must have \[ \square \] to the left and \[ \square \] to the right. \[ \square \] must have \[ \square \] above and to the left of it. \[ \square \] must have \[ \square \] above and to the right of it. \[ \square \] must have \[ \square \] below it and to its left. \[ \square \] must have \[ \square \] below it and to the right of it. \[ \square \] can be adjacent to anything but \[ \square \] except as noted above. \[ \square \] can be adjacent to anything but \[ \square \] except as noted above. These rules imply that inside the rectangle delineated by the \[ \square \], \[ \square \], \[ \square \], and \[ \square \] tiles are only \[ \square \] tiles, and immediately outside it are \[ \square \] tiles. Each rectangle is outlined by \[ \square \], \[ \square \], \[ \square \], and \[ \square \] tiles on the sides, and \[ \square \], \[ \square \], \[ \square \], and \[ \square \] on the corners, and is full of \[ \square \] inside. A vertical stripe has \[ \square \] on its left and \[ \square \] on its right, and a horizontal stripe has \[ \square \] above it and \[ \square \] below it. Both vertical and horizontal stripes have only \[ \square \] inside them. The rectangles and stripes cannot be adjacent and are separated by \[ \square \] tiles.

**Interactions between layers 1 and 2:** A layer 2 \[ \square \] tile must be in the same spot as a layer 1 \[ \square \] or \[ \square \], and only a \[ \square \] tile can be in those spots on layer 2. The \[ \square \] and \[ \square \] layer 2 tiles must be in the same spot as a layer 1 \[ \square \] tile, but a layer 1 \[ \square \] tile could also be in the same spot as a layer 2 \[ \square \] tile. A layer 1 \[ \square \] tile *cannot* be in the same spot as any other layer 2 tile.

Once we look at the compatibility conditions between the two layers, we see that in fact layer 2 can only contain rectangles which must be lined up with the space between the horizontal and vertical lines on layer 1. On layer 1, each rectangle delineated by the horizontal and vertical lines must have a \[ \square \] in the upper left corner, which starts a diagonal line of \[ \square \] tiles extending towards the bottom right. Since it must end at the border of the rectangle, but a \[ \square \] tile cannot be in the same spot as a layer 2 \[ \square \] or \[ \square \] tile, the only place the diagonal line can end is at the lower right corner. Thus, the rectangle must actually be a square. (See figure 6 for an example of an allowed layer 1 tiling.)

However, the only way for all the rectangles formed by the horizontal and vertical layer 1 lines to be squares is if the spacing between them is equal. They then form \( M \times M \) squares arrayed in a \( k \times k \) grid for a total of \( k^2 \) squares. It follows that \( N = k(M + 1) \). But when \( N \) is prime, then \( k \) must be 1. (\( M \) cannot be 0 since the horizontal and vertical lines cannot be adjacent.) Thus, the only allowed tiling is to produce a single \( (N - 1) \times (N - 1) \) square on layer 1. We can then consider some data layers of tiles which implement the rules from section 5. The layer 2 \[ \square \], \[ \square \], \[ \square \], and \[ \square \] tiles mark out the corner of the square, so we can put a condition on the data layers that enforce the corner boundary conditions on those locations.

We do need to be careful of one aspect, however, since we are now restricted to a size of square which is 1 less than a prime number. For any input \( x \), we need to choose a prime \( N \). \( N \) should be not much bigger

---

Figure 6: A possible arrangement of layers 1 (a.) and 2 (b.) for PERIODIC TILING.
than \( x (\log N = \text{poly}(\log x)) \), and it must be possible for the universal TM implemented by the data layers to deduce \( x \) in a reasonable time. It is known that there exists a constant \( \theta < 1 \) such that

\[
\lim_{y \to \infty} \left[ \pi(y + y^\theta) - \pi(y) \right] = \frac{y^\theta}{\log y},
\]

where \( \pi(y) \) is the number of primes less than or equal to \( y \) \([\text{Ing37}]\). (For our purposes, it is sufficient to take \( \theta = 2/3 \), but smaller \( \theta \) is possible.) That is, for sufficiently large \( y \), the number of primes in the interval \([y, y + y^\theta]\) is approximately \( 1/\log y \) times the size of the interval

Thus, given \( x \), we will let \( n_0(x) = 2\lceil \log x \rceil \) (that is, basically twice the number of bits in \( x \)) and let \( N_0(x) = x 2^{n_0(x)} \) (that is, the binary expansion of \( N_0(x) \) is that of \( x \) followed by \((n_0(x) - 1) \) 0s). Then

\[
N_0(x)^\theta = x^\theta 2^{\theta n_0(x)} \leq 2^{\theta(\lceil \log x \rceil + n_0(x))} \leq 2^{n_0(x)}.
\]

In particular, it follows that \( N_0(x) + N_0(x)^\theta \leq N_0(x + 1) \). If we can find a prime \( N \) in the interval \( I(x) = [N_0(x), N_0(x) + N_0(x)^\theta) \), the TM can thus easily deduce \( x \) by looking at the most significant bits. There is at least one prime in \( I(x) \), so we can certainly find one with an exhaustive search, which can be done with polynomial space. Furthermore, by (4), if we choose a random \( N \in I(x) \), for sufficiently large \( x \), there is a probability about \( 1/\log(N_0(x)) \) that \( N \) is prime, which we can verify in \( \text{poly}(\log x) \) time. Thus, we get a randomized reduction to PERIODIC TILING which runs in expected time \( \text{poly}(\log x) \).

### 5.2 Weighted Constraints

Now we consider the case where the constraints count different amounts.

**Definition 5.3** WEIGHTED TILING

**Problem Parameters:** A set of tiles \( T = \{t_1, \ldots, t_m\} \). 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. An integer \( c \).

**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 \( c \).

Notice that in this case we do not make any restrictions on the boundary conditions. We can define an additional variant of WEIGHTED TILING where we instead have periodic boundary conditions (the top and bottom rows are considered adjacent, as are the left-most and right-most columns).

The definition of WEIGHTED TILING is not completely analogous to the quantum 1-DIM TIH problem. We require the maximum cost to be a constant independent of \( N \). More generally, we could allow \( c \) to be a polynomial in \( N \). This does not change the complexity of any case for which we have been able to prove a definite result, with the exception of WEIGHTED TILING with reflection symmetry and periodic boundary conditions, where we have shown the problem is in \( \mathbb{P} \) when \( c \) is constant, and have no result when \( c = \Omega(N) \).

**Theorem 5.4** WEIGHTED TILING is \( \text{NEXP}-\text{complete} \), as is the periodic boundary conditions version of WEIGHTED TILING.

**Proof:** WEIGHTED TILING with open boundary conditions: It is sufficient to take all the weights to be 0, +2, or −1, and \( c = −4 \). We will reduce from TILING; the main point is to show we can fix the boundary conditions on the corners. To do so, we take the TILING instance and add a new layer of tiles

\[\text{This is the result one might expect from the Prime Number Theorem, but that theorem is not strong enough, as it is compatible with having a large interval with a low density of primes.}\]
Figure 7: The preferred arrangement of the new layer in WEIGHTED TILING with open boundary conditions.

consisting of five special types of tile, \(\boxdot\), \(\boxcirc\), \(\boxless\), \(\boxgreater\), and \(\boxtimes\). will have a weight +2 if it is below any type of tile at all, and a weight +2 if it is to the left of any type of tile at all. However, it has a weight −1 if it is to the right of any type of tile. Therefore, if \(\boxtimes\) is located anywhere but the top right corner, it will provide a net cost of at least +1, but if it is in the top right corner, it provides a net benefit: −1 to the total cost. Similarly, for \(\boxless\), \(\boxgreater\), and \(\boxcirc\) we arrange rules that cause them to be preferentially in the top left, bottom left, and bottom right corners, each providing −1 to the total cost when in the correct positions, and at least +1 elsewhere. The pairing of \((\boxless, \boxcirc)\) in either the vertical or horizontal directions has a weight 0. In the main layer of tiles, we use only the weights 0 and +2, with +2 indicating a pairing that would be forbidden in the original TILING problem and 0 indicating a pairing that would be allowed. Finally, we insist that \(\boxdot\), \(\boxcirc\), or \(\boxless\) in the special layer must correspond to the usual corner tile \(t_1\) in the main layer. Then in order for there to be a solution with total cost at most −4, the new layer must in fact consist of \(\boxless\), \(\boxgreater\), \(\boxcirc\), and \(\boxtimes\) at the corners, and \(\boxcirc\) elsewhere in the new layer (as in figure 7), and the main layer is constrained in exactly the way it would be in the standard TILING problem.

**WEIGHTED TILING with periodic boundary conditions:** We can of course apply Theorem 5.2, but when we allow weights, there is a simpler solution which avoids the caveats about the reduction.

We now consider only odd \(N\), not necessarily prime \(N\). We will use the weights 0, +1, and +3, and set \(c = +2\). Thus, to have a good enough tiling, we can have two pairings with weight +1, and all the others must have weight 0.

As above, we add a new layer, again with five types of tile, \(\boxless\), \(\boxgreater\), \(\boxcirc\), \(\boxtimes\), and \(\boxdot\). We assign a weight 0 to any pairing with \((\boxless, \boxcirc)\) adjacent, either vertically or horizontally and with either orientation. However, we assign a weight +3 to have \((\boxless, \boxcirc)\) or \((\boxgreater, \boxdot)\) adjacent vertically or horizontally. Thus, the preferred arrangement is for \(\boxless\) and \(\boxcirc\) to tile the plane with a checkerboard pattern. Of course, they cannot do that completely when \(N\) is odd.

We allow \(\boxless\) to be above or below \(\boxcirc\) or \(\boxdot\) (weight 0 for such pairings), and we allow \(\boxcirc\) to be to the left or right of \(\boxless\) or \(\boxdot\) (again, assign weight 0 to any such pairing). To the right or left of \(\boxcirc\) we have weight 0 to have another \(\boxcirc\) but above or below \(\boxcirc\) we have a cost of +3 to have \(\boxless\) or \(\boxdot\). Similarly, above or below \(\boxcirc\) we have weight 0 to have \(\boxcirc\), but we have weight +3 to have \(\boxless\) or \(\boxdot\) to the right or left of \(\boxcirc\) Thus, \(\boxcirc\) will preferentially form horizontal lines, and \(\boxcirc\) will preferentially form vertical lines.

Finally, we have weight 0 to have \(\boxless\) above or below \(\boxcirc\) and weight +1 to have \(\boxcirc\) to the right or left of \(\boxless\). We have a weight +3 to have \(\boxless\) to be adjacent to \(\boxcirc\) or \(\boxdot\) in any direction. Therefore, the only place \(\boxless\) can be to provide total cost +2 is at the intersection of a horizontal line of \(\boxless\)’s and a vertical line of \(\boxcirc\)’s, and there is no place it can go to have a cost of 0 or +1. The cost of having a line of \(\boxless\)’s and \(\boxcirc\)’s intersect without a \(\boxcirc\) is even higher, as is the cost of not having a line of \(\boxless\)’s or \(\boxcirc\)’s at all. The upshot is that any tiling of the new layer must be mostly a checkerboard of \(\boxless\) and \(\boxcirc\) with exactly one horizontal line of \(\boxless\)’s and one vertical line of \(\boxcirc\)’s, which intersect in a \(\boxdot\). Those horizontal and vertical lines will determine the boundary of an \((N - 1) \times (N - 1)\) grid, and we can set the boundary conditions at the corners via adjacency to the \(\boxcirc\) and \(\boxtimes\) tiles. The arrangement is much like that of figure 6a, but with \(\boxless\) replaced by \(\boxcirc\) and \(\boxcirc\) replaced by \(\boxtimes\).
5.3 One-Dimensional Tiling

In one dimension, the tiling problem becomes the following:

Definition 5.5 1-DIM TILING
Problem Parameters: A set of tiles $T = \{t_1, \ldots, t_m\}$. A set of 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 designated tile $t_1$ that must be placed at the ends of the line.

Problem Input: Integer $N$, specified in binary.
Output: Determine whether there is a valid tiling of a line of length $N$.

We can also define a WEIGHTED 1-DIM TILING problem analogously to the WEIGHTED TILING problem in section 5.2.

Theorem 5.6 1-DIM TILING and WEIGHTED 1-DIM TILING are in P.

Proof: Unweighted case: Let us create a directed graph with $m$ nodes. The $i$th node corresponds to the $i$th tile type $t_i$, and there is an edge between $i$ and $j$ iff $(t_i, t_j) \in H$. We wish to know whether there is a path of length exactly $N$ starting and ending at $t_1$. Certainly this can be done in time $\text{poly}(N)$, but we actually wish to do it in time $\text{polylog}(N)$. We will also generalize slightly to allow the left end of the line to have tile $t_0$ and the right end to have tile $t_1$.

First, let us make a table of all paths which start and ending at the same node and contain no additional cycles. Such a path is called a simple path. A path is a simple path if it is a simple cycle or it contains no cycles. Let $M$ be the set of all simple cycles, and let $l(p)$ be the length of the path $p$. Clearly when $p$ is a simple path, $l(p) \leq m + 1$, which is constant, so constructing the table of simple cycles takes constant time.

Given any path $p$, we can decompose it into a simple path and a multiset $P$ of simple cycles by progressively removing simple cycles from $p$ until we are left with a simple path, adding each removed cycle to $P$.

Definition 5.7 A multiset $P$ of simple cycles is allowed for the simple path $p$ if there exists a path $q$ which can be decomposed as above into exactly the multiset $P$ plus $p$.

Claim 5.8 The multiset $P$ is allowed iff the underlying set $Q$ for $P$ is allowed.

Proof of claim: If we have any path through the node $t_i$, we can insert a simple cycle for $t_i$ to get another possible path. If $q$ is a path which can be decomposed to the set $Q$, then every node on every simple cycle in $P$ appears in $q$, so we can insert as many copies as we need of the simple cycles in $P$.

Conversely, let $q$ be a path for the allowed multiset $P$. We can express $q$ as a tree, with the root being the simple path $p$ and each node being a simple cycle. Cycle $c'$ is a child of $c$ if they share a node and $c'$ is not the parent of $c$. The tree can be obtained by iteratively taking two consecutive occurrences of a node $t_i$ along a cycle such that there are no repetitions between the occurrences of $t_i$ and identifying the two occurrences of $v$. We wish to modify the tree so that the resulting path $q'$ has the same multiset $P$ as its decomposition, but with the property that there is a subtree rooted at $p$ containing just one of every simple cycle in $Q$. Then we can take the path $q$ corresponding to that subtree, and $q$ has the set $Q$ for its decomposition.

That is, we need that within the tree, for every $r \in Q$, there exists an occurrence $c$ of $r$ such that the path from $p$ to $c$ in the tree does not pass through two occurrences of any simple cycle. Since we can move any node corresponding to a simple cycle for $t_i$ to be a child of any simple cycle that contains $t_i$, we can put the tree in the desired format by moving one instance of each $r \in Q$ up the tree until it lies off of the first cycle which contains $t_i$.

Claim 5.9 There is a tiling of a line of length $N$ from $t_0$ to $t_1$ iff $\exists$ simple path $p$, $f : M \to \mathbb{Z}^+ \cup \{0\}$ such that:
1. \( p = (t_{i_0}, t_{i_1}, \ldots, t_{i_L}) \), with \( i_0 = 0, i_L = 1, l(p) = L + 1 \),

2. \( L + 1 + \sum_{q \in M} f(q)l(q) = N \),

3. \( f^{-1}(\mathbb{Z}^+) \) is allowed for \( p \).

Given an allowed multiset \( P \) of simple cycles, the function \( f(q) \) represents the number of times the particular simple cycle \( q \) appears in the multiset, which means that \( f^{-1}(\mathbb{Z}^+) \) is the underlying set for \( P \). To see what Claim \( 5.9 \) is true, observe that the resulting path contains \( N \) nodes, as desired. Conversely, given \( p \) and \( f \), we can create an allowed multiset \( P \) for \( p \) by including each simple cycle \( q \) a number of times equal to \( f(q) \).

There are only a constant number of simple paths \( p \) and allowed sets \( P \) of simple cycles. By running over \((p, P)\) and applying Claim \( 5.9 \) the 1-DIM TILING problem reduces to determining if there exists \( f' : P \rightarrow \mathbb{Z}^+ \cup \{0\} \) such that \( \sum_{q \in P} f'(q)l(q) = n' \), \( n' = n - (L + 1) - \sum_{q \in P} l(q) \). (We take \( f'(q) = f(q) - 1 \), with \( f \) given by the claim.) With fixed \((p, P)\), we must therefore solve the following problem: Given a set of \( m' \) positive integers \( a_k \) (the set of distinct \( l(q) \) for \( q \in P \)), do there exist non-negative integers \( b_k \) (the corresponding \( f'(q) \), summed over any cycles with equal lengths) such that \( \sum_k a_kb_k = N' \)? This is a special case of the Unbounded Knapsack Problem \([\text{Lue75}]\) where the set of objects is fixed and only the total cost allowed varies.

Note that, if we allowed \( b_k \) to be negative, this would just be answered by determining if \( N' \) is a multiple of \( \gcd(a_1, \ldots, a_{m'}) \). Indeed, if \( N' \) is greater than \( m'g \), where \( g \) is the least common multiple of the \( a_k \), this is all we need to determine: If \( \gcd(a_1, \ldots, a_{m'}) \mid r \), then we can write \( r = \sum_k a_kc_k \), with the property that \( a_kc_k \geq -g \) for all \( k \). Thus, if \( N' = m'g + r \), then \( N' = \sum_k a_kb_k = N' \), providing our solution. Conversely, if \( \gcd(a_1, \ldots, a_{m'}) \not\mid N' \), then it is not possible that \( \sum_k a_kb_k = N' \). For \( N' < m'g \), matters are more complicated, but since \( g \leq m! \) is a constant, we can answer those cases via a look-up table.

**Weighted case:** The 1-DIM TILING problem remains easy even if we take a weighted variant. In this case, the directed graph we get is a complete graph, but the edges have a weight associated with them, and we want to determine the minimal cost path of length \( N \) from \( t_{i_0} \) to \( t_{i_1} \). Now we need to count both the length and the cost of each simple cycle. That is, we need to learn if \( \exists \) simple path \( p, f : M \rightarrow \mathbb{Z}^+ \cup \{0\} \) such that:

1. \( p = (t_{i_0}, t_{i_1}, \ldots, t_{i_L}) \), with \( i_0 = 0, i_L = 1, l(p) = L + 1 \),

2. \( c(p) + \sum_{q \in M} f(q)c(q) = c \),

3. \( f^{-1}(\mathbb{Z}^+) \) is allowed for \( p \).

where \( c(q) \) is the cost of path \( q \).

Once we fix \((p, P)\), we again get a set \( a_k \) of positive integer lengths \( l(q) \) for \( q \in P \), but we also have \( c_k \), the costs \( c(q) \) of \( q \in P \). We can assume \( f'(q) = 0 \) for all but one cycle \( q \in P \) for each length — we should pick the one with the lowest cost \( c(q) \). Now we wish to find non-negative integers \( b_k \) that minimize \( \sum_k b_kc_k \) subject to \( \sum_k a_kb_k = N' \), which is again a special case of the Unbounded Knapsack Problem. We now have simple cycles of every length up to \( m \), so \( m' = m \), and we again set \( g \leq m! \) to be the least common multiple of the \( a_k \) (i.e., the lcm of \( 1, \ldots, m \)). We create a look-up table to give an optimal path for \( N < mg \). When \( N \) is a multiple of \( g \), note that we can find an optimal path by simply calculating which simple cycle has the lowest ratio \( c_k/a_k \); then we just use that cycle \( g/a_k \) times.

For other values of \( N' = dq + r \), with \( d \geq m \), we just look at the optimal path for \((m-1)g + r \) and add \((d - m + 1)g/a_k \) copies of the optimal simple cycle. This will be an optimal path for \( N' \). To show this, note that we can assume without loss of generality that an optimal solution will not have \( b_k \geq g/a_k \) for more than one value of \( k \): Suppose there were two values \( k_1, k_2 \) for which \( b_k \geq g/a_k \), with \( c_{k_1}/a_{k_1} \leq c_{k_2}/a_{k_2} \).

Then we can shift \( b_{k_1} \mapsto b_{k_1} + g/a_{k_1} \) and \( b_{k_2} \mapsto b_{k_2} - g/a_{k_2} \), leaving the total length the same without
increasing the cost. Furthermore, using the same logic, we know that the one value \( k_1 \) for which \( b_k \geq g/a_k \) must have a minimal value of \( c_k/a_k \). For \( N! > mg \), there must actually exist such a value of \( k_1 \), so the optimal path must have been formed in the way we have described.

\[ \text{5.4 Additional Symmetry} \]

We consider two additional kinds of symmetry. 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 \( (t_i, t_j) \in V \) 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.

Note that these types of reflection and rotation symmetries assume that we can reflect or rotate the tiling rules without simultaneously reflecting or rotating the tiles. For instance, if a tile \( t_i \) has a pattern on it (such as \[ \text{Figure 8} \]) that looks different when it is turned upside down, then when we reflect vertically, we also could reflect the tile, producing a new tile \( R(t_i) \) (\[ \text{Figure 8} \] in this case). We could define a reflection symmetry for this type of tile too: \( (t_i, t_j) \in H \) iff \( (R(t_j), R(t_i)) \in H \), etc., but we just get the same complexity classes as for the case with no additional reflection symmetry. This is because we can add an extra layer of tiles with arrows on them and put on a constraint that any adjacent arrow tiles must point the same direction. While either direction will work (or any of the four directions in the case of rotation symmetry), one direction ends up preferred in any given potential tiling, so by looking at the arrow tile in a given spot, we can effectively reproduce rules that have no reflection symmetry. Thus, we address the case where the rules have reflection or rotation symmetry without simultaneously reflecting or rotating the tiles.

**Theorem 5.10** When the constraints for TILING have reflection symmetry, there exist \( N_e, N_o \in \mathbb{Z}^+ \cup \{\infty\} \) such that for even \( N \geq N_e \) or odd \( N \geq N_o \), a valid tiling exists, while for even \( N < N_e \) and odd \( N < N_o \), there is no tiling (except for \( N = 1 \), when there is a trivial tiling).

- When we have open boundary conditions, either \( N_e = N_o = \infty \) or \( N_e = N_o = 1 \).
- When we have the four corners boundary condition, either \( N_o = 3 \) or \( N_e = N_o = \infty \).
- When we have periodic boundary conditions, either \( N_e = 2 \) or \( N_e = N_o = \infty \).
- When we have rotation symmetry, \( N_e \) and \( N_o \) are computable.

When we have reflection symmetry but not rotation symmetry, we have been unable to determine so far whether \( N_e \) and \( N_o \) are computable for the four corners and periodic boundary conditions, respectively. Note that we do not consider just one corner fixed for the boundary condition, since that would break the reflection symmetry.

**Proof:** To prove the theorem, simply note that given a valid tiling of an \( N \times N \) grid, with \( N \geq 4 \), we can extend it to a valid tiling of an \( (N + 2) \times (N + 2) \) grid. The main observation is that we can repeat existing patterns when we have reflection symmetry, because if \( AB \) is a legal configuration, so is \( ABAB \).

In order to extend a tiling, we can do the following: strip off the leftmost column and bottommost row, and replace them with duplicates of the next two rows and columns. We can fill in the corner by duplicating the bottom left \( 2 \times 2 \) square once we have stripped off the rows. Then the original leftmost column and bottommost row became the new leftmost and bottommost column and row, with some duplication to lengthen them to the right size. (See figure \[ \text{8} \])

This strategy handles even more general boundary conditions than the three main cases we consider. In particular, it also works if the tiling rules on the sides of the grid are completely different from the tiling rules.
Figure 8: Extending a tiling of a $6 \times 6$ grid to $8 \times 8$ when the tiling conditions have a reflection symmetry. The spaces on the right mark the repeated rows and columns.

Figure 9: Extending the tiling of a corner to a tiling of a $3 \times 3$ grid with fixed corner boundary conditions. in the interior of the grid. When the tiling rules are the same on the sides as in the center, except perhaps for the corners, we can copy the outermost two rows and columns, so this strategy works for $N \geq 2$.

**Four corners boundary conditions:** When $N$ is odd and we have the four-corners boundary condition, we can look to see if there is a tiling of a $2 \times 2$ grid with the corner tile in the upper right corner. If so, we can duplicate the right column to add a column on the left, and then duplicate the top row to add a row on the bottom. (See figure 9.) This gives us a tiling of the $3 \times 3$ grid with all four corners correctly tiled. Conversely, if there is no $2 \times 2$ tiling containing one of the corner tiles, then there cannot be a tiling of any $N \times N$ grid with $N > 1$, which means that $N_o = N_e = \infty$.

**Open boundary conditions:** In the case of completely open boundary conditions, we need only check a $2 \times 2$ grid to see if there is a valid tiling. If not, there cannot be a tiling of any size $N > 2$ either. If there is, we can extend it as in figure 9 to get a tiling of the $3 \times 3$ grid as well, and extend as in figure 8 to get a tiling of any size grid.

**Periodic boundary conditions:** When we have periodic boundary conditions, a valid tiling of a $2 \times 2$ grid with open boundary conditions gives us a valid periodic tiling of a $2 \times 2$ grid as well, so we can extend it to all even $N$. However, if we try to apply the strategy of figure 9 to extend it to odd $N$, we potentially ruin the periodicity, so we do not know if $N_o$ is computable in this case. If there is no tiling of a $2 \times 2$ grid with open boundary conditions, then there can not be a tiling for any $N$ with periodic boundary conditions and $N_o = N_e = \infty$.

**Rotation symmetry:** When we have rotation symmetry (with any boundary conditions), it will suffice to compute $N_e$ and $N_o$ as the minimum even and odd lengths that allow us to tile a single side of the square. If a tiling of one side exists, we can use this same tiling on all four sides and then fill in the center using diagonal stripes of identical tiles, as in figure 10. In the case where there are special rules for the boundary, we will need to tile a side plus the adjacent row/column as preparation, but this presents little additional difficulty.

To tile a single side, we can use an approach similar to the previous 1-dimensional case. However, now matters are much simpler, since the graph is now undirected. There are thus always many size 2 cycles, so we need only find the minimal even- and odd-length cycles for $t_1$. That sets an upper bound on $N_e$ and $N_o$. It might be one of these can be made smaller, but that is straightforward to check as well.

The weighted case is more difficult, and currently, when there is only reflection symmetry, we only have
a result for the case of periodic boundary conditions. We will show that something similar to theorem 5.10 holds in this case as well.

**Theorem 5.11** For WEIGHTED TILING with additional symmetry, we have the following results:

- With reflection symmetry and periodic boundary conditions: There exist $N_e, N_o \in \mathbb{Z}^+ \cup \{\infty\}$ such that for even $N \geq N_e$ or odd $N \geq N_o$, a valid tiling exists, while for even $N$, $2c < N < N_e$ and odd $N$, $2c < N < N_o$, there is no tiling. $N_e$ is computable.

- With rotation symmetry and open boundary conditions: Either there exists computable $N_0 \in \mathbb{Z}^+$ such that tiling is possible for $N \geq N_0$ and impossible for $N < N_0$ or there exists computable $N_0 \in \mathbb{Z}^+$ such that tiling is possible for $N < N_0$ and impossible for $N \geq N_0$. ($N_0$ depends on the weights and maximum allowed cost $c$.)

- With rotation symmetry and periodic boundary conditions: One of the following three cases holds: There exists computable $N_0 \in \mathbb{Z}^+$ such that tiling is possible for all $N \geq N_0$, there exists computable $N_0 \in \mathbb{Z}^+$ such that tiling is impossible for all $N \geq N_0$, or tiling is possible for all even $N$ and there exists computable $N_o \in \mathbb{Z}^+ \cup \{\infty\}$ such that for odd $N \geq N_o$, tiling is possible.

- With rotation symmetry and the four-corners boundary condition: There exist computable $N_e, N_o \in \mathbb{Z}^+ \cup \{\infty\}$ such that either there exists a tiling for any even $N \geq N_e$ or for no even $N \geq N_e$, and either there exists a tiling for any odd $N \geq N_o$ or for no odd $N \geq N_o$.

Currently, we do not know if the $N_o, N_e$ for the periodic boundary conditions with reflection symmetry are computable.

**Proof:** Reflection symmetry and periodic boundary conditions: Let us suppose we have a tiling of the $N \times N$ grid with periodic boundary conditions with total cost $c' \leq c$. Let the rows of this tiling be $R_1, \ldots, R_N$, and because of the periodic boundary conditions, let $R_{N+1} = R_1$. Let $w(R_i) = \sum w(t_a, t_b)$, where the sum is taken over adjacent tiles $t_a$ and $t_b$ in row $R_a$ and let $w(R_i, R_{i+1}) = \sum w(t_a, t_b)$, where the sum is now taken over adjacent tiles $t_a$, $t_b$, where $t_a$ is in row $R_i$ and $t_b$ is in row $R_{i+1}$. Then $c' = \sum_i (w(R_i) + w(R_i, R_{i+1}))$. Notice that if we duplicate two adjacent rows, say $R_i$ and $R_{i+1}$, then the total cost becomes $c' + w(R_i) + w(R_{i+1}) + 2w(R_i, R_{i+1})$ (because of the reflection symmetry). Now,

$$\sum_{i=1}^{N} [w(R_i) + w(R_{i+1}) + 2w(R_i, R_{i+1})] = 2 \sum_{i=1}^{N} [w(R_i) + w(R_i, R_{i+1})] = 2c'. \quad (6)$$

Thus, there must exist some $i$ for which

$$w(R_i) + w(R_{i+1}) + 2w(R_i, R_{i+1}) \leq 2c'/N. \quad (7)$$

When $c'$ is a constant, for $N > 2c'$, that means there is some $i$ for which $w(R_i) + w(R_{i+1}) + 2w(R_i, R_{i+1}) \leq 0$, since the weights are integers. Then we can duplicate rows $R_i$ and $R_{i+1}$ without increasing the cost.
Similarly, there are two columns which we can duplicate without increasing the cost. Thus, we also have a tiling for the \((N + 2) \times (N + 2)\) square grid with cost \(c'' \leq c' \leq c\). The only difference from theorem 5.10 is that there might be some small exceptions with \(N \leq 2c\) which have valid tilings but cannot be extended. \(N_e\) then is the smallest even value for \(N\) such that \(N_e \geq 2c\) and there is a valid tiling of the \(N_e \times N_e\) grid of weight \(c' \leq c\). Similarly, \(N_o\) is the smallest value for \(N\) such that \(N_o \geq 2c\) and there is a valid tiling of the \(N_o \times N_o\) grid of weight \(c' \leq c\).

Note that this argument requires that the cost \(c\) be a constant. It fails if \(c = c(N)\) grows at least linearly with \(N\).

When \(N\) is even, we can easily compute the exact minimal cost achievable. Consider all possible \(2 \times 2\) squares of tiles, and compute the cost of each by adding the costs of the four adjacent pairs in the square. Given any possible tiling of the periodic \(N \times N\) grid of total cost \(c\), let \(S_{a,b}\) be the \(2 \times 2\) square whose top left corner is in location \((a,b)\), and let \(w(S_{a,b})\) be the cost of \(S_{a,b}\). Then \(\sum_{a,b} w(S_{a,b}) = 2c\). Therefore, the minimal possible cost achievable for the \(N \times N\) grid is \(N^2 w/2\), where \(w\) is the minimal cost of any possible \(2 \times 2\) square. When \(N\) is even, this is actually achievable by repeating a minimal cost square \(N/2\) times in each direction. The squares with their top left corner in location \((a,b)\), with \(a, b\) even are exactly the square that we chose to repeat, and the squares in other locations are reflections of that square, which have the same cost.

Rotation symmetry and open boundary conditions: This case is essentially trivial. We find the minimal weight between any pair of tiles, and simply tile the square in a checkerboard pattern with those two tiles. That is certainly the minimum cost achievable.

Rotation symmetry and periodic boundary conditions: Find the lowest weight of a cycle of length \(N\) for the one-dimensional WEIGHTED TILING problem. Call that \(w\). The minimum achievable total cost for \(2\)-D WEIGHTED TILING with rotation symmetry and periodic boundary conditions is then \(2Nw\) (\(N\) rows and \(N\) columns each of cost \(w\)). This can actually always be done using the same tiling as the unweighted rotation symmetry case (figure 10). Thus, the problem reduces to the one-dimensional case, which we have argued is in \(P\). However, with the additional reflection symmetry, the argument can be simplified. If all the weights are positive, then \(w \geq N\) and tiling is only possible for small \(N\). If there exists a negative pair of weights, then for sufficiently large \(N\), the minimal cost path will use many of that negative pair, and tiling is always possible for sufficiently large \(N\). If the smallest weight is 0, then any sufficiently long one-dimensional cycle with constant weight \(w\) must use a 0-cost pairing, and it is possible to duplicate the pair of tiles used in that 0-cost pairing. Thus, the minimum weight cycle of length \(N + 2\) is also at most \(w\). Therefore, we need only determine the minimum odd \(N\) for which \(w = 0\), which is the length of the smallest odd-length cycle involving only zero-cost pairings. (When \(N\) is even and there is a zero-cost pairing, a zero-cost cycle is always possible by alternating between the two tiles involved in the pairing.)

Rotation symmetry and 4-corners boundary condition: This case is more difficult, but still computationally easy (in \(P\)). For large \(N\), the idea is that we will again tile most of the square using a pair of tiles \(t_i, t_j\) with minimal cost to be adjacent. That determines the asymptotic scaling of the total cost for large \(N\). However, the details are more complicated because of the effect of the corners.

First, note that if the minimum weight for any pair of tiles is positive, then it is certainly not possible to tile an \(N \times N\) grid when \(2N(N - 1) > c\). Conversely, if there is a pair of tiles \((t_i, t_j)\) such that the cost for having them adjacent is negative, then we can always tile an \(N \times N\) grid for sufficiently large \(N\) by taking a checkerboard of \(t_i\) and \(t_j\), with only the corners different (if the designated corner tile \(t_1\) is not already \(t_i\) or \(t_j\)). Suppose that \(w(t_1, t_i), w(t_1, t_j) \leq w\). Then “sufficiently large” \(N\) means \(N\) such that \(2N(N - 1) - 8 \geq 8w - c\).

The only difficult case is thus when there are pairs of tiles \((t_i, t_j)\) with \(w(t_i, t_j) = 0\), but no pairs with a negative cost. Consider the graph whose nodes are tile types \(t_i\) and edges for any pairing of tiles with 0 cost. Let \(Z_\alpha\) be a set of tiles comprising a connected component of this graph with more than one tile, and let \(Z = \cup_\alpha Z_\alpha\). We will show that any tiling of total cost at most \(c\) must consist mostly of tiles from one particular \(Z_\alpha\), with only a constant number of different tiles, and that indeed it is sufficient to place all the
non-$Z_\alpha$ tiles near the corners of the grid. Then by considering all possible low-cost combinations of tiles near the corners, we will be able to determine the minimum achievable cost for the grid.

We will call the \textit{k-square for the upper left corner} to be the $k \times k$ square of tiles in the upper left corner of the grid. The \textit{upper-left k-border} consists of the rightmost column and bottom row of the $k$-square for the upper left corner. We make similar definitions for the other corners: the upper-right $k$-border is the leftmost column and bottom row of the $k$-square for the upper right corner, the lower left $k$-border is the rightmost column and top row of the $k$-square for the lower left corner and the lower right $k$-border is the leftmost column and top row of the $k$-square for the lower right corner. An example is given in Figure \ref{fig:11}. We say that a square is \textit{valid} if its border contains only zero-cost adjacent pairs and the corner of the square that is also the corner of the larger grid has the correct corner tile. Since a $k$-border must be connected that means that the edges within the border are all contained within a particular connected component $Z_\alpha$.

We will determine if there is an $N \times N$ tiling of the grid by considering all valid tilings for the squares in each corner of the grid where we allow the sizes of the squares to go up to $2c$. For each such combination, we require that the total cost of the tiling be some $c' \leq c$ and that the tiles in each border come from the same connected component $Z_\alpha$. We then must determine whether it is possible to fill the remainder of the grid with zero-cost adjacent pairs, possibly interrupted by a few additional adjacent pairs with total cost at most $c - c'$ (although it will turn out that this is never necessary). Suppose we have such a tiling, and consider tile $t_i \in Z_\alpha$ on the border of one square, and tile $t_j \in Z_\alpha$ on the border of a different square. Then, as shown below, there must exist a path from $t_i$ to $t_j$ that only includes zero-cost adjacent pairs of tiles. Note that the parity of the length of the path does not depend on which path we choose.

More generally, given two $k$-squares $I_1$ and $I_2$ for different corners of the grid, we are interested in whether it is possible to create a path of zero-cost adjacent pairs to get from $t_i \in I_1$ to $t_j \in I_2$, without necessarily being concerned about whether we can tile the whole grid:

\textbf{Definition 5.12} Suppose we have two $k$-squares $I_1$ and $I_2$ that are located at different corners of the grid, with borders from $Z_\alpha$. We say they are compatible for $N$ and $Z_\alpha$ if, given tile $t_i$ on the border of $I_1$ and tile $t_j$ on the border of $I_2$, there is a path of zero-cost pairings from $t_i$ to $t_j$ of the correct length for $I_1$ and $I_2$ to be placed together on an $N \times N$ grid. We say a set of four squares $I_1$, $I_2$, $I_3$, and $I_4$ which fit into the four corners of the grid is compatible for $N$ and $Z_\alpha$ if they are compatible in pairs.

Note that it does not matter which tile we pick from the border to define compatibility. This is because if we use $t_i'$ on the border of $I_1$ instead of $t_i$, we can take a path from $t_i'$ to $t_i$ using only the zero-cost pairings in the border of $I_1$, and concatenate it with the path from $t_i$ to $t_j$. Similarly, if $I_1$, $I_2$, and $I_3$ are valid squares located at three different corners, then if $I_1$ is compatible with $I_2$ and $I_2$ is compatible with $I_3$, it follows that $I_1$ is compatible with $I_3$. Also, if $I_1$ and $I_2$ are compatible for $N$, they are also compatible for $N + 2$: Because of the reflection symmetry, we can repeat a pair of adjacent tiles in a path to lengthen it by 2.

For large $N$, if there is a tiling of the $N \times N$ grid with total cost at most $c$, then there must exist, for some $\alpha$, valid squares in the four corners $I_1$, $I_2$, $I_3$, and $I_4$ which are compatible for $N$ and $Z_\alpha$ such that the total cost of $I_1$, $I_2$, $I_3$, and $I_4$ is at most $c$ and the sizes of the squares are between $c$ and $2c$. To see why this is true, note that there can be only $c$ non-zero-cost adjacent pairs in the entire tiling. Therefore, as $k$ ranges from $c$ to $2c$, it must be the case that for each corner at least one $k$-border contains only zero-cost adjacent

![Figure 11](image_url)
Figure 12: Extending a square to make a corner triangle. The black tiles have nonzero cost to be adjacent to any other tile. All other pairings that occur in the figure are zero cost.

pairs. Furthermore, for each such pair of borders, there are at least $c + 1$ disjoint paths from a tile in one border to a tile in the other border. This follows from the fact that the borders themselves have more than $c$ tiles. At least one of these paths must contain only zero-weight pairings. Thus we know that for sufficiently large grids, if a there is a tiling of total weight at most $c$, then there must be four valid squares for the four corners which are compatible for $N$ and $Z_\alpha$. The converse is also true:

Lemma 5.13 Let $I_1, I_2, I_3,$ and $I_4$ be four squares which are compatible for $N$ and $Z_\alpha$, $N$ sufficiently large. Then there is a tiling of the $N \times N$ grid using $I_1, I_2, I_3,$ and $I_4$ in the four corners, with only tiles from $Z_\alpha$ used in the rest of the grid.

Proof of lemma: Assume without loss of generality that $I_1$ is in the upper left corner, $I_2$ is in the upper right corner, $I_3$ is in the lower left corner, and $I_4$ is in the lower right corner of the grid.

We will extend each square to create an isosceles right triangle whose outer diagonal border is composed of just one kind of tile. This can be done by progressively duplicating the outer sides of the square, each time shifting by one space towards the edge of the grid, as in figure 12. Finally, we can add additional diagonal layers to make sure the outer diagonal border is any particular tile from $Z_\alpha$, and to increase the size of the triangle as much as desired.

We can define compatibility of the newly created corner triangles in the same way as for the squares. When squares $I$ and $J$ for two different corners are compatible for $N$ and $Z_\alpha$, then their corresponding triangles remain compatible when extended as described above. In particular, there is a path of zero-cost tile pairs that will let us tile the top row of the grid between $I_1$ and $I_2$. Then we can copy the tiling of the interval between triangles diagonally down and to the left, much as in figure 10 describing the unweighted case. This fills in the upper left corner of the grid, up to the diagonal line defined by the leftmost tile of the triangle for $I_2$ and the topmost tile of the triangle for $I_3$. We can do the same thing for the lower right corner of the grid, tiling the right edge between $I_2$ and $I_4$, and copying this tiling diagonally down and to the left. This leaves only a diagonal strip between $I_2$ and $I_3$, and we can tile that using a checkerboard pattern consisting of the outermost tile $t_i$ for the triangles and any other tile from $Z_\alpha$ which has a zero-cost pairing with $t_i$. See figure 13 for an example of a complete grid tiled in this way.

Thus, we can use the following algorithm to determine what the lowest achievable cost is for large even and odd grid sizes, when the lowest-cost pairings are cost 0: List the low-cost valid squares for each corner up to size $2c$ and determine which sets of four are compatible for large even $N$ and for large odd $N$. The lowest total cost for each case tells us the minimum achievable cost.

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 QMA$_{\text{EXP}}$ respectively.

We have considered a number of variants of the classical 2-dimensional tiling problems, but there are a few cases in which we have not been able to determine the difficulty. In particular, when we have additional reflection symmetry (but not rotation symmetry), weighted tiling remains open with open boundary conditions or boundary conditions at the corners. In a number of the other cases with reflection or rotation symmetry, we know the problem is easy, but do not know whether the algorithm we have suggested involves an uncomputable parameter. Another very interesting question is the difficulty of the quantum 1-DIM TIH problem with reflection symmetry, or for that matter, the quantum 2-DIM TIH problem with reflection or rotation symmetry. We have no results for those cases.

Acknowledgements

We would like to thank John Preskill for pointing out the connection to $N$-REPRESENTABILITY, and Cris Moore and Igor Pak for some information about classical tiling problems. 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 Grant CCR-0514082. Part of this work was done while both authors were visiting the Institute for Quantum Information at Caltech.

References

[AGIK07] Dorit Aharonov, Daniel Gottesman, Sandy Irani, and Julia Kempe. The power of quantum systems on a line. In FOCS, pages 373–383, 2007. arXiv:0705.4077v2 [quant-ph].

Figure 13: A 16 × 16 grid tiled by extending triangles on the four corners.
[BDCG89] 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*, pages 204–216, New York, NY, USA, 1989. ACM.

[Ber66] R. Berge. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 66:1–72, 1966.

[Boa97] Peter Van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, pages 331–363. Marcel Dekker Inc, 1997.

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26:11–20, 1997.

[Deu85] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Royal Society of London Proceedings Series A*, 400:97–117, July 1985.

[Ing37] A. E. Ingham. On the difference between consecutive primes. *Quarterly Journal of Mathematics (Oxford Series)*, 8:255–266, 1937.

[Ira09] S. Irani. Ground state entanglement in one dimensional translationally invariant quantum systems, 2009. arXiv:0901.1107v1 [quant-ph].

[JWZ07] Dominik Janzing, Pawel Wocjan, and Shengyu 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].

[Kay08] Alastair Kay. The computational power of symmetric hamiltonians. *Phys. Rev. A*, 78:012346, 2008. arXiv:0801.3228v3 [quant-ph].

[KSV02] A.Y. Kitaev, A.H. Shen, and M.N. Vyalyi. *Classical and Quantum Computation*. AMS, Providence, RI, 2002.

[Liu07] Y.-K. Liu, M. Christandl, and F. Verstraete. \( N \)-representability is QMA-complete. *Phys. Rev. Lett.*, 98:110503, 2007. arXiv:quant-ph/0609125.

[Lev86] Leonid A Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986.

[LP97] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation (2nd Edition)*. Prentice Hall, 2 edition, August 1997.

[Luc75] George Lueker. Two NP-complete problems in nonnegative integer programming. *Computer Science Laboratory, Princeton University*, 1975. Report No. 178.

[NW08] Daniel Nagaj and Pawel Wocjan. Hamiltonian quantum cellular automata in 1D, 2008. arXiv:0802.0886v4 [quant-ph].

[OT05] R. Oliveira and B. Terhal. The complexity of quantum spin systems on a two-dimensional square lattice, 2005. quant-ph/0504050.

[Pap95] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1995.

[Wan60] Hao Wang. Proving theorems by pattern recognition I. *Commun. ACM*, 3(4):220–234, 1960.