THEORY OF COMPUTING, Volume 9 (2), 2013, pp. 31–116
www.theoryofcomputing.org

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

Daniel Gottesman† Sandy Irani‡

Received July 25, 2011; Revised October 1, 2012; Published January 25, 2013

Abstract: We study the complexity of a class of problems involving satisfying constraints which remain the same under translations in one or more spatial directions. In this paper, we show hardness of a classical tiling problem on an $N \times N$ 2-dimensional grid and a quantum problem involving finding the ground state energy of a 1-dimensional quantum system of N particles. In both cases, the only input is N, provided in binary. We show that the classical problem is NEXP-complete and the quantum problem is $\mathsf{QMA}_{\mathsf{EXP}}$-complete. Thus, an algorithm for these problems which runs in time polynomial in N (exponential in the input size) would imply that $\mathsf{EXP} = \mathsf{NEXP}$ or $\mathsf{BQEXP} = \mathsf{QMA}_{\mathsf{EXP}}$, respectively. Although tiling in general is already known to be NEXP-complete, the usual approach is to 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.

ACM Classification: F.1.3

AMS Classification: 68Q17, 81P68

Key words and phrases: tiling complexity, quantum Hamiltonian complexity

*A preliminary version of this paper was posted on the arXiv in 2009 [12]. An extended abstract appeared in FOCS'09.

†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.

‡Supported in part by NSF Grant CCR-0916181. Part of this work was done while both authors were visiting the Institute for Quantum Information at Caltech.

DOI: 10.4086/toc.2013.v009a002

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 [23, 3], in which one can show that "typical" cases of some language are hard. For example, [23] gives an example of a tiling problem which is NP-complete on average. In this paper we take a different approach. In many cases, practically interesting instances possess some shared property, such as a symmetry, that distinguish them from the general instance and might, in principle, make those instances easier. We will study such an example and show that, even in a system possessing a great deal of symmetry, it is still possible to prove a hardness result.

Specifically, we consider the related problems of determining whether there is a possible tiling of an r-dimensional grid with some fixed set of classical tiles and of finding the lowest energy state (or *ground state*) of a quantum system involving interactions only between neighboring particles on an r-dimensional grid. The ground state energy of a system is considered one of the basic properties of a physical system, and over the last few decades, physicists have developed a number of heuristics that have been successful in finding the ground state energy in many special cases. On the other hand, in earlier work [1, 2], we have shown that in the most general case, even in a 1-dimensional quantum system, finding the ground state is a computationally difficult problem (modulo the usual complexity-theoretic assumptions). However, the construction presented in [1] involves a system which is completely unnatural from a physical point of view. The most interesting physical systems frequently possess an additional symmetry: translational invariance. In this paper, we will show that even a 1-dimensional translationally-invariant system can be hard.

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

1.1 The tiling problem

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

$\mathsf{NEXP} = \mathsf{EXP}$. Note that $\mathsf{NEXP} \neq \mathsf{EXP}$ is a stronger complexity assumption than $\mathsf{P} \neq \mathsf{NP}$ although both assumptions are widely believed to hold. $\mathsf{P} = \mathsf{NP}$ implies that $\mathsf{NEXP} = \mathsf{EXP}$ although the converse is not necessarily true. This can be seen by a standard padding argument (see [29] for this proof and a more in-depth discussion of these complexity classes).

This version of tiling is equivalent to the more common Wang Tiles [34] 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 many prior results showing that related tiling problems are NEXP-complete. Most of these reductions require that either the set of tiles and their constraints or some varying boundary conditions be given as part of the input [25, 9, 24, 13, 10]. 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.

The Tile Assembly Model introduced by Winfree [35] is an augmentation of Wang's tiling model in which the rules specify how a tiling can be constructed with local rules. In this model, each pair of tiles has a bond strength when they are placed next to each other. A new tile can be placed on the boundary of a partial tiling if it forms bonds whose bond strengths sum to a given threshold value. It has been shown that this model is Turing-universal. The input to a problem to be solved still must be encoded in an initial set of boundary constraints or the rules and tile types. There has been a significant research effort to implement this model in molecular system, in particular, to create systems that construct a desired shape. Theoretical effort has been concerned with the efficiency (i. e., number of tiles) required to produce a particular shape [32].

Our interest in tiling was motivated initially by quantum Hamiltonian complexity, in which translationally-invariant systems are physically very natural, but previously there was no apparent way to prove hardness results about such systems. Meanwhile, although the study of tiling is quite old by computer science standards, it was not until the last few years that researchers had examined similar issues in the context of tiling. Our proof that translationally-invariant tiling is NEXP-complete is also proven independently as a result on *picture languages* in [6] in a paper which originally appeared in a 2007 conference. For completeness, we include the proof of the theorem here expressed as a tiling problem as it lays the groundwork for ideas used later in the paper for the quantum case and other generalizations of tiling.

Many NEXP-complete problems are succinct versions of familiar combinatorial problems [11, 30] in which the input has some special structure which allows for a more compact representation. For example, consider the problem of finding an independent set in a graph where the graph is specified by indicating the number of nodes in binary and providing a compact rule (or circuit) to determine if two nodes are connected. Traditionally, the rule is included as part of the input, which potentially allows for a more expressive language. The analog to our work would be for the rule to have a constant-sized description, fixed for all inputs. Philosophically similar approaches have been taken by Valiant [33] and Jones and Selman [18]. Jones and Selman study the set of cardinalities for which a fixed first-order formula has a model, and Valiant considers counting objects (for instance graphs) of a variable size N with a fixed rule (such as containing a certain fixed set of subgraphs).

The basic idea of our construction is to reduce from an instance x of some language in NEXP by encoding x in the binary expansion of N, the size of the grid. It is well known that a finite set of tiling rules can be used to implement a universal Turing Machine. We need some way to express the program for the Turing Machine to run, and that program must grow with the size of x. Previous constructions managed this by resorting to either polylog N different tile types or varying boundary conditions to encode x, but those are both fixed, constant-sized parameters in our version of the problem. Instead, we use the tiles to implement a binary counter which converts N into binary and then uses it as an input to a universal Turing Machine. The idea of encoding a counting process in a tiling system appeared as early as 1984 in [10], but in the tiling problems considered there, the tiling constraints are still considered part of the input.

1.2 Hamiltonian ground state complexity

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.

1.2.1 Relationship between tiling and Hamiltonians

It is worth being more explicit about the relationship between tiling problems and Hamiltonian problems. Tiling problems essentially correspond to Hamiltonians for classical spin systems. In a classical spin system, each particle has a classical state lying in a set T. We are interested in the case where T is finite. The particles have interactions among sets of particles, and the total energy is the sum of the energies from each interaction. This can be written as a Hamiltonian which is diagonal in the standard basis.

Every tiling problem can be written as a classical spin system by choosing T to be the set of tile types and letting the diagonal Hamiltonian term between a pair of particles be 0 if the pairing of tiles is allowed and 1 if it is not allowed. Any ground state of such a system is a classical state in which the state of each particle is specified by one of the possible standard basis states in T. A pair of tiles (t_i, t_j) is allowed by the tiling rules iff the corresponding $|t_i t_j\rangle\langle t_i t_j|$ term of the Hamiltonian is 0, so that allowed tilings have 0 total energy, whereas a forbidden tiling has energy at least 1. Therefore, the question of whether a globally allowed tiling exists is the same as the question whether this Hamiltonian has a ground state energy of 0.

We can also write an arbitrary classical spin system with finite state space T as a tiling problem, but to do so, we should generalize the notion of tiling to have weighted constraints between tiles. Now the Hamiltonian terms connecting sets of particles can have any energy values, not just 0 and 1. To make a corresponding tiling problem, we assign a weight for each possible set of tiles. The total cost of a tiling is then the sum of the weights given by the local terms. We thus ask the question, does there exist a tiling whose total cost is less than or equal to some function $p(N)$? By allowing $p(N) > 0$, we consider the possibility of *frustration*: The ground state energy of the spin system may not be 0, and we are interested in how much greater than 0 it is.

Classical spin systems are much studied, and since such spin systems need not have all energy terms be 0 or 1, the notion of weighted tiling is a very natural one from a physics point of view. Translational invariance is a common feature, as for quantum spin systems, as are additional symmetries such as reflection or rotation invariance. Thus, it is interesting to study the complexity of weighted tilings with those symmetries as well as unweighted ones.

As a concrete example, consider the classical Ising model in 1 dimension. The Ising model uses particles with two spin states and has a Hamiltonian of the form $H = \sum_i J_i Z_i Z_{i+1}$. The term $Z_i Z_{i+1}$ has eigenvalues $+1$ and -1. If the spin states are 0 and 1, then $+1$ eigenvalue corresponds to even parity (00 or 11) and -1 eigenvalue is odd parity (01 or 10). When $J_i = -1$, the energy of the term is minimized when the spins are the same; this is equivalent to a model with two tile types t_0 and t_1 with the requirement that the adjacent tiles at locations i and $i+1$ be the same. When $J_i = +1$, the energy of the term is minimized when the spins are opposite, which corresponds to a tiling model which forbids the same type of tile at locations i and $i+1$. As we have written the Hamiltonian, the terms are not positive semi-definite so the ground state energy is actually less than 0, but we can easily shift the energy to make it 0: $H = \sum_i (J_i Z_i Z_{i+1} + 1)$. We can also let J_i take values other than ± 1, which corresponds to a weighted tiling problem, with weight J_i for an adjacent pair for which the tiles are the same and weight $-J_i$ when they are different.

1.2.2 The local Hamiltonian problem

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 therefore 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 defined a decision problem based on finding the ground state energy of a local Hamiltonian as follows: there exist two values $a > b$, such that $a - b \geq 1/\mathrm{poly}(N)$, where it is guaranteed that the ground state energy is at most b or at least a, and one wants to determine only which of the two alternatives holds. He also introduced the class QMA, the quantum analog of NP, and showed that the local Hamiltonian problem is QMA-complete [22].[1] Thus, we do not hope to solve it even on a quantum computer. The

[1] Technically, this problem is "Promise QMA-complete," since we must be given a promise about the ground state energy

problem is still hard even for two-dimensional systems on qubits or one-dimensional systems of particles of constant Hilbert space dimension [28, 1].

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

Since highly symmetric systems are rather natural, a number of researchers have studied the computational power of translationally invariant quantum systems. For instance, [27] gives a 20-state translation-invariant modification of the construction from [1] (improving on a 56-state construction by [16]) that can be used for universal 1-dimensional adiabatic computation. In adiabatic computation, a quantum system in its ground state is slowly deformed into a new one for which the ground state encodes the solution to a computational problem. This construction requires that the system be initialized to a particular configuration in which each particle is in a state that encodes some additional information, in this case the input to the problem to be solved. There is a ground state associated with each possible way to initialize the system. 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 [20] 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 [15]. While quantum entanglement is closely related to the performance of numerical heuristics in practice, the particular states in this construction are easy to compute.

In contrast, we show that there exist 1-dimensional translationally-invariant quantum systems with nearest-neighbor interactions for which finding the ground state energy is complete for $\mathsf{QMA}_{\mathsf{EXP}}$. The class $\mathsf{QMA}_{\mathsf{EXP}}$ is the quantum analogue of NEXP, where the witness is a quantum state and the verifier is a quantum circuit. 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. This size of the input is $\log N$, so an algorithm to solve our ground state problem in time that is polynomial in N, the number of particles in the system, would imply that $\mathsf{QMA}_{\mathsf{EXP}} = \mathsf{BQEXP}$, where BQEXP is like BQP, but with exponential circuits. As in the classical case, $\mathsf{QMA}_{\mathsf{EXP}} \neq \mathsf{BQEXP}$ is a stronger assumption than $\mathsf{QMA} \neq \mathsf{BQP}$, but both are widely believed to hold. To our knowledge, this is the first appearance of the

before we know that it is in QMA. However, it and other such problems are commonly described as "QMA-complete" nevertheless. There are no known problems which are QMA-complete without the promise. In this paper, we use the common terminology.

class $\mathsf{QMA_{EXP}}$ in the literature. 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.

One consequence of our result is that it is now possible to talk about the hardness of a specific Hamiltonian term rather than the hardness of a class of Hamiltonians. Since a system with a computationally difficult Hamiltonian cannot find its own ground state, it is likely that such a system will behave like a spin glass at low temperatures. The usual models of spin glasses have randomly chosen coefficients, causing a breakdown of translational invariance. The systems we construct in this paper are different, with a completely ordered, translationally-invariant Hamiltonian, even though the ground states are quite complicated. Any disorder in the system is emergent rather than put in by hand, a property that these spin glasses would share with structural glasses. Some other systems with "self-induced disorder" have been introduced previously, although not in the context of computational complexity [7]. In all of our hardness constructions, we construct specific two-particle terms designed to let us prove that the resulting Hamiltonian problems are hard, but one can imagine going the other direction, and studying the complexity of a particular Hamiltonian term given to you. However, we currently have no techniques for doing so.

It is worth noting that the one-dimensional version of the classical tiling problem is very easy: it is in P (see Section 4.3 for the algorithm). That is, it can be solved in a time polylog N, whereas it appears the quantum problem takes time $\exp(N)$, even on a quantum computer (unless $\mathsf{QMA_{EXP}} = \mathsf{BQEXP}$). Translational invariance does seem to simplify the 1-dimensional classical case, reducing poly(N) time to polylog(N) time, but it doesn't help very much in the quantum case.

1.3 Structure of the paper

Section 2 contains the technical definitions of the main problems we consider and summarizes our results. Sections 3 and 4 contain the results on classical tilings, and Sections 5, 6, 7, and 8 contain the results on quantum Hamiltonians.

Section 3 contains the basic translationally-invariant tiling result and is a necessary prerequisite for the study of the variants of tiling considered in Section 4. Many of the variant results are independent of each other, however, and it should be largely possible to focus on only those proofs the reader finds interesting. However, some techniques are re-used in later parts of the section, particularly the analysis of tilings in one dimension (Subsection 4.3).

For the quantum sections, it is helpful, but not essential, to have read Section 3. While the details of the proof for the quantum case are different from those of the proof for classical tiling, the basic structure of the construction is similar, so an understanding of the classical tiling case is useful for following the quantum construction. Section 5 contains the basic quantum result, with open boundary conditions on a finite chain. A general understanding of Section 5 is necessary for reading Sections 6 (for periodic boundary conditions), 7 (which adds a reflection symmetry), and 8 (which discusses hardness for a problem with infinitely many spins). However, not all details of the proof in Section 5 are needed in the later sections.

2 Problems and results

The paper contains a variety of different but related results involving classical tiling and quantum Hamiltonian problems with translational invariance. In this section, we will summarize the different variants, giving the proofs and more detailed discussion of each variant in later sections. While there are certain recurring techniques, the details of the different cases vary considerably.

2.1 Tiling 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: Positive integer N, specified in binary.
Output: Determine whether there is a valid tiling of an $N \times N$ grid.

Theorem 2.2. *There exists a (T, H, V, t_1) for which TILING is* NEXP*-complete.*

A similar comment holds for all the hardness results given in this paper. Although Theorem 2.2 is subsumed by a result proven independently in [6], for completeness, we give the proof in Section 3. The basic idea is that the corner tiles 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. The set of colors for the interior tiles is the cartesian product of the layer 1 colors and the layer 2 colors. The rules for the two layers are independent except at the boundary. Thus, in the interior of the grid, two tiles can be placed next to each other as long as the placement is consistent with the layer 1 colors for the tiles (according to the layer 1 rules) and the layer 2 colors for the tiles (according to the later 2 rules). In the construction, each layer 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 $\Theta(N^{1/k})$ for some constant 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 non-deterministic Turing machine which computes a NEXP-complete language. The rules for the top boundary check whether the final configuration on layer 2 is an accepting state. [6] goes about their proof somewhat differently although the end result is the same. They look at a generalization of strings called *pictures* which are essentially matrices over a finite alphabet. *Picture languages* are sets of these objects. In particular, they study the complexity of recognizing unary picture languages whose elements are squares. They show that such a language corresponds to the set of tilable squares according to a finite set of tiling rules if and only if the language defined by the binary representation of the sizes of the squares is in NEXP time. This proof uses a binary counter similar to our construction. In a separate proof, they show that finite tiling systems can simulate an arbitrary Turing Machine. It turns out that the width of the grid is not important in this construction as long as it is an integer at least N.

Note that it is important that we chose to have the input N provided in binary. If it were instead given in unary, there would only be one instance per problem size, and the problem would be trivially in P/poly.

Thus, in order to prove a meaningful hardness result, we are forced to move up the exponential hierarchy and prove the problem is NEXP-complete rather than NP-complete.

Notice that in our basic TILING problem we specify the tile in all four corners of the grid. One might wonder if this requirements can be weakened. As we show in Section 4.1, if we specify just one corner (as is common in the literature, e. g., [29]) or no corner at all, the resulting tiling problem becomes trivial. This does not happen if the tiling rules are part of the input (as is assumed in [29]) since the rules can then encode the action of different Turing Machines starting from a single tile. If, instead, we specify two or three corners the problem remains NEXP-complete even with fixed tiling rules.

Thus, the boundary conditions are a critical component. As well as fixing the tiles at the 4 corners of the square, we have considered periodic boundary conditions (so we are actually tiling a torus) and open boundary conditions, where there are no additional constraints on the tiles that can be placed on the boundary of the square beyond those given by H and V. 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.

Subsequent to the preliminary version of our paper [12], a similar result about periodic tilings was proven independently in [17]. However, their results are slightly different than ours. They are asking whether there exists a periodic tiling with a specific period n. We prove that determining whether there is a tiling of the $n \times n$ torus is hard, which not only requires showing that there is a tiling with period n for the "yes" instances, but also requires showing that in the "no" instances, the torus cannot be broken up into smaller regions whose size is a factor of n. Our construction effectively disallows breaking up the torus in this way by restricting the inputs to prime n. Thus, our proof implies that the periodic tiling problem is hard. However, this is accomplished at the expense of a weaker reduction as the proof is accomplished by a randomized poly-time reduction in order to find a prime in the required range.

We have also considered problems with additional symmetry beyond the translational invariance. If we have *reflection symmetry*, then if $(t_i,t_j) \in H$, then $(t_j,t_i) \in H$ as well, and if $(t_i,t_j) \in V$, then $(t_j,t_i) \in V$ also. That is, the tiling constraints to the left and right are the same, as are the constraints above and below. However, if we only have reflection symmetry, there can still be a difference between the horizontal and vertical directions. If we have *rotation symmetry*, we have reflection symmetry and also $(t_i,t_j) \in H$ iff $(t_i,t_j) \in V$. Now the direction does not matter either. These additional symmetries are well motivated from a physical point of view since many physical systems exhibit reflection or rotation symmetry. Finally, we have studied the one-dimensional version of the problem as well as the two-dimensional version. See Table 1 for a summary of our results. Proofs are given in Section 4.

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 decide 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.

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

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.

2.2 Quantum results

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

Definition 2.3. A language L is in $\mathsf{QMA}_{\mathsf{EXP}}$ iff there exists a k and a Quantum Turing Machine M such that for each instance x and any $|\psi\rangle$ on $O(2^{|x|^k})$ qubits, on input $(x, |\psi\rangle)$, M halts in $O(2^{|x|^k})$ steps. Furthermore,

(a) if $x \in L_{\text{yes}}$, $\exists |\psi\rangle$ such that M accepts $(x, |\psi\rangle)$ with probability at least $2/3$;

(b) if $x \in L_{\text{no}}$, then $\forall |\psi\rangle$, M accepts $(x, |\psi\rangle)$ with probability at most $1/3$.

A quantum system consists of a set of n dimensional particles. The set of states of the individual particles is a d-dimensional Hilbert space for some constant d and the set of states for the entire system is a d^n-dimensional Hilbert space. The *Hamiltonian* is the energy operator for the system which is a Hermitian operator that acts on this Hilbert space. While the matrix representation of the Hamiltonian is a $d^n \times d^n$ matrix, most Hamiltonians of physical interest are *k-local*, meaning that they can be expressed as a sum of terms, each of which operates nontrivially on only k particles. Since there can be at most n^k such terms, these Hamiltonians have more succinct representations. We focus on an even more restricted (but still physically motivated) class of Hamiltonians in which particles are arranged on a grid, all terms operate on nearest neighbor pairs, and the terms do not depend on the location of the pair (but may depend on their orientation relative to each other).

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: Positive integer N, specified in binary.
Promise: Consider an N^r-dimensional grid of particles and the Hamiltonian resulting from applying H_i to each pair of neighboring particles along dimension i. The ground state energy of this system is either at most $p(N)$ or at least $p(N)+1/q(N)$.
Output: Determine whether the ground state energy of the system is at most $p(N)$ or at least $p(N)+1/q(N)$.

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

Theorem 2.5. *There exist (H_1, p, q) for which 1-DIM TIH is $\mathsf{QMA_{EXP}}$-complete.*

The proof of Theorem 2.5 is given in Section 5. It immediately implies that r-DIM TIH is $\mathsf{QMA_{EXP}}$-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 5.

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 t^{th} state has the form $|\phi_t\rangle|t\rangle$, where $|\phi_t\rangle$ is the t^{th} state in the process we wish to simulate, and the overall ground state will be $\sum_t |\phi_t\rangle|t\rangle$. The size of the system controls the number of time steps for which the clock runs. In the case of the construction presented here, the process consists of two main phases. The first phase is the execution of a Turing machine which simply increments a binary counter. The clock ensures that this TM is run for $N-3$ steps after which a number that is $\Theta(N^{1/k})$ for some constant k is encoded in binary in the state of the quantum system. This state is then used as the input to an arbitrary quantum Turing machine which is executed in the second phase. This QTM implements a verifier which is also allowed a quantum witness of length $\Theta(N)$. Finally, there is an energy term which penalizes any non-accepting computation of the verifier.

We can also consider variants of 1-DIM TIH. If we use periodic boundary conditions instead of open boundary conditions, we get the same result (see Section 6). If we add reflection symmetry, the problem also remains $\mathsf{QMA_{EXP}}$-complete with open or periodic boundary conditions (see Section 7).

1-DIM TIH is more closely analogous to WEIGHTED TILING rather than the unweighted version because we allow frustration in the ground state. For this reason, open boundary conditions are sufficient for hardness. One could define an unfrustrated version of 1-DIM TIH with $p(N) = 0$, but it is a bit less natural than the classical unweighted TILING problem because it is not possible to check efficiently if a state has 0 energy or simply an exponentially small energy.

As a corollary of Theorem 2.5, the following version of N-REPRESENTABILITY [26] is also $\mathsf{QMA_{EXP}}$-complete: Given a density matrix ρ on two d-state particles, is it within ε of a state ρ' such that there exists a translationally-invariant pure state $|\psi\rangle$ for N particles arranged in a circle for which ρ' is the marginal state of two adjacent particles? ρ is a parameter of the problem, and N, given in binary, is the only input, as in our Hamiltonian problem. We can reduce to this version of N-REPRESENTABILITY by starting with 1-DIM TIH on a circle. Then there is always a translationally-invariant pure ground state $|\psi\rangle$ of the Hamiltonian H. By breaking the Hilbert space of two d-state particles up into small balls, we can get a finite set of density matrices ρ to try. For each one, if we can solve N-REPRESENTABILITY, we can determine if ρ can be extended to a candidate ground state $|\phi\rangle$, and if so we can determine the energy of $|\phi\rangle$, since it is just equal to $N \cdot \mathrm{tr}(H_1 \rho)$. Trying all possible ρ, we can thus find the ground state energy of H, up to some ε-dependent precision.

Another case of particular physical interest is the infinite chain. Since it may be the case that the energy of the ground state of a system will grow with the length of the chain, an infinite chain will not necessarily have a well-defined energy. For this reason, we examine the energy per particle in the ground state. This is defined as follows. Given the ground state, we can measure the energy of that state over a finite segment by restricting ourselves to the energy contribution from the local terms contained within the segment. The energy per particle within that segment is the energy of the segment divided by the number of particles. The energy per particle of the infinite chain is obtained by taking the limit of this quantity as the segment grows in both directions. Of course, if the Hamiltonian term is fixed and the chain is infinite, the ground state energy of the system is a single number and there is not a computational problem with an infinite family of inputs to study. Instead, we look at a variation where the two-particle Hamiltonian term is the input to the problem and the size of the input is the number of bits required to specify the term. We then ask: if this term is applied to each pair of neighboring particles in an infinite chain, what is the ground energy per particle?

Definition 2.6. ITIH (Infinite-Translationally-Invariant Hamiltonian)
Problem Parameter: Three polynomials p, q, and r. d, the dimension of a particle.
Problem Input: A Hamiltonian H for two d-dimensional particles, with matrix entries which are multiples of $1/r(N)$.
Promise: Consider an infinite chain of particles and the Hamiltonian resulting from applying H to each pair of neighboring particles. The ground state energy per particle of this system is either at most $1/p(N)$ or at least $1/p(N) + 1/q(N)$.
Output: Determine whether the ground state energy per particle of the system is at most $1/p(N)$ or at least $1/p(N) + 1/q(N)$.

We prove the following theorem:

Theorem 2.7. *There exist (p,q,r,d) for which ITIH is $\mathsf{QMA_{EXP}}$-complete.*

In the body of this paper, we use a shorthand for the statements of theorems since we are dealing with parametrized families of languages: when we say a problem is NEXP-complete or $\mathsf{QMA_{EXP}}$-complete, we actually mean there exists a set of parameters for which the language is NEXP-complete or $\mathsf{QMA_{EXP}}$-complete. Thus, Theorem 2.7 would become "ITIH is $\mathsf{QMA_{EXP}}$-complete."

3 Hardness of TILING

We note that all of the problems considered here are in NEXP since a valid tiling can be specified in $2^{O(n)}$ bits and verified in time that is linear in the specification of the tiling. For the result of the section,

Tile on left \ Tile on right	⊞	◧	◨	⬒	⬓	$*$
⊞	N	N	N	Y	Y	N
◧	N	N	N	N	N	
◨	N	N	N	N	N	N
⬒	Y	N	N	Y	N	N
⬓	Y	N	N	N	Y	N
$*$	N	N		N	N	

Tile on bottom \ Tile on top	⊞	◧	◨	⬒	⬓	$*$
⊞	N	Y	Y	N	N	N
◧	Y	Y	N	N	N	N
◨	Y	N	Y	N	N	N
⬒	N	N	N	N	N	N
⬓	N	N	N	N	N	
$*$	N	N	N		N	

Table 2: The tiling rules for boundary tiles. $*$ represents any interior tile. "N" indicates a disallowed pairing of tiles. For two boundary tiles, "Y" represents an allowed pairing. Some of the rules for interior tiles are not specified, because they depend on the specific interior tile.

therefore, we focus on proving hardness. The variants in Section 4 are also in NEXP by the same argument.

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 $f_{BC} : \mathbb{Z} \to \{0,1\}^*$ such that for some constant N_0 and every $N \geq N_0$, if M_{BC} runs for N steps, then the string $f_{BC}(N)$ will be written on the tape with the rest of the tape blank. Moreover there are constants c_1 and c_2 such that if n is the length of the string $f_{BC}(N)$ and $N \geq N_0$, then $2^{c_1 n} \leq N \leq 2^{c_2 n}$. We will also assume that for any binary string x, we can compute N such that $f_{BC}(N) = x$ in time that is polynomial in the length of x. In some of the variations of the problem we consider in Section 4, we will need to put additional restrictions on N (such as requiring N to be odd), and in those cases, we still require that we can find an N with the appropriate restrictions such that $f_{BC}(N) = x$.

Using a standard padding argument, we can reduce any language in NEXP to $\mathsf{NTIME}(2^{c_1 n})$. If L is in $\mathsf{NTIME}(2^{n^k})$, the reduction consists of padding an input x so that its length is $|x|^k/c_1$ [29]. Thus, we will take an arbitrary non-deterministic Turing machine M which accepts an NEXP-complete 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. The Turing Machine M_{BC} will be run for $N-3$ steps. The -3 comes from the fact that the $N \times N$ grid holds two boundary rows as well as the starting row. This will leave the string x on the final row of M_{BC}'s computation.

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 ⊞. (We actually only need to use two corners as described in Section 4.1.) First we will specify a set of boundary tiles and their constraints. In addition to ⊞ there are four other kinds of boundary tiles: ◧, ⬒, ⬓, ◨. We will call the rest of the tiles *interior* tiles. The tiling rules for the boundary tiles are summarized in Table 2.

◧ will mark the left side of the grid, ⬒ the top of the grid, ⬓ the bottom of the grid, and ◨ the right side of the grid. (See Figure 1.) To show that in any valid tiling the boundary tiles must be placed in this

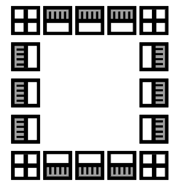


Figure 1: The only allowed tiling of the sides of a 5×5 grid.

manner, note the following facts:

- Nothing can go to the left of a tile which means that the only place a tile could go is the left-most boundary.
- Similarly, tiles can only go in the top row, tiles can only go in the bottom row, and tiles can only go in the right-most column.
- No interior tile can border a tile in any direction. Furthermore a cannot border on itself in any direction. This means that the only possible locations for a are the four corners since those are the only places which can be surrounded by , , , or tiles. Since the boundary conditions state that tiles must go in the corners, those are exactly the locations that will hold tiles.
- The only tiles that can go above or below a tile are and tiles, and cannot go on the west boundary. Thus, the tiles on the west boundary adjacent to the corners must be tiles.
- Only or can go above or below a tile, so the entire west boundary, except for the corners, will be tiles.
- Similar logic shows that the entire east boundary, except for the corners, will be tiles. Also, the entire south boundary, except for the corners, will be tiles, and the entire north boundary, except for the corners, will be tiles.

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

Although it is well known that tiling rules are Turing complete [19], 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.

		Tile on right					
		$[b]$	$[b,q',r]$	$[b,q',l]$	$[b,q',R]$	$[b,q',L]$	◨
	$[a]$	Y	Y	N	Y	N	Y
Tile	$[a,q,r]$	N	N	N	N	If $q=q'$	N
on	$[a,q,l]$	Y	N	N	N	N	Y
left	$[a,q,R]$	N	N	If $q=q'$	N	N	N
	$[a,q,L]$	Y	N	N	N	N	Y
	◧	Y*	Y	If $q'=q_0$	Y	N	Y

		Tile on top				
		$[b]$	$[b,q',r]$	$[b,q',l]$	$[b,q',R]$	$[b,q',L]$
	$[a]$	If $a=b$	If $a=b$	If $a=b$, $q'\neq q_0$	N	N
Tile	$[a,q,r]$	N	N	N	If TM rule	If TM rule
on	$[a,q,l]$	N	N	N	If TM rule	If TM rule
bottom	$[a,q,R]$	If $a=b$	If $a=b$	If $a=b$, $q'\neq q_0$	N	N
	$[a,q,L]$	If $a=b$	If $a=b$	If $a=b$, $q'\neq q_0$	N	N

Table 3: The tiling rules to simulate a Turing machine. "N" indicates a disallowed pairing of tiles, and "Y" represents an allowed pairing. Pairings with "if" statements are allowed only if the condition is satisfied. "If TM rule" means the pairing is allowed only if $(a,q)\rightarrow(b,q',L/R)$ is one of the valid non-deterministic moves of the Turing machine being simulated. The "Y*" entry will be modified later to get the correct starting configuration for the Turing machine.

The non-deterministic Turing machine M is specified by a triplet (Σ,Q,δ), with designated blank symbol $\#\in\Sigma$, start state $q_0\in Q$ and accept state $q_A\in Q$. There are three varieties of tiles, designated by elements of Σ (variety 1), $\Sigma\times Q\times\{r,l\}$ (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. The $\{r,l\}$ symbol in a variety 2 tile tells us from which direction the head came in its last move. 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.

The tiling rules are given in Table 3. 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 from the bottom row to the middle row and the move $(c,q')\rightarrow(f,q'',R)$ from the middle row to the top row:

$[f,q'',R]$	$[b,q'',l]$	$[d]$
$[c,q',r]$	$[b,q',L]$	$[d]$
$[c]$	$[a,q,r]$	$[d,q,L]$

The lower row shows the head in the square with the a. The $[d,q,L]$ is from the previous TM move. The tile $[b,q',L]$ in the middle row enforces that the tiler is committing to executing the step $(a,q)\rightarrow(b,q',L)$, although there may have been other possible non-deterministic choices. The $[c,q',r]$ tile to the left of the $[b,q',L]$ shows the new location and state of the head after the first move. The $[b,q',L]$ tile now just acts as a $[b]$ tile for purposes of the tiling above. The tile $[f,q'',R]$ enforces that the

Additional layer 1 rules:

Boundary Tile Location	Adjacent interior tile $[a]$	$[a,q,r]$	$[a,q,l]$	$[a,q,R]$	$[a,q,L]$
Bottom	Y	Y	Y	Y	Y
Top	If $a = \#$	N	If $a = \#$ and $q = q_0$	N	N
Left	If $a \neq \#$	Y	If $q = q_0$	Y	N

Additional layer 2 rules:

Boundary Tile Location	Adjacent interior tile $[a]$	$[a,q,r]$	$[a,q,l]$	$[a,q,R]$	$[a,q,L]$
Bottom	If a matches layer 1	N	If $q = q_0$ and a matches layer 1	N	N
Top	Y	If $q = q_A$	If $q = q_A$	Y	Y
Left	If $a \notin \Sigma_{M_{BC}}$	Y	If $q = q_0$	Y	N

Table 4: Additional tiling rules between certain boundary and interior tiles for layers 1 and 2. Note that these rules are specific to the boundary only because we have established that the boundary tiles can only be placed at the boundary. For layer 2, the alphabet symbol $a \in \Sigma$ must match the alphabet symbol for the corresponding layer 1 tile when on the bottom interior row.

tiler is committing to executing the step $(c,q') \rightarrow (f,q'',R)$. The $[b,q'',l]$ tile to the right of the $[f,q'',R]$ shows the new location and state of the head after the second move.

To be consistent with the horizontal tiling rules, a row must consist of some number of variety 1 tiles, along with adjacent pairs consisting of a variety 2 tile and a variety 3 tile. The variety 2 and 3 tiles must be matched in the following sense: $q = q'$, and either we have r on the left and L on the right or R on the left and l on the right. At the west edges of the row, we could possibly have just a single variety 2 tile by itself. However, this can only happen if the tile is of the form $[a,q,l]$. The rules enforce then that a tile can only go to the left of a variety 2 tile if the Turing Machine state is q_0, which is the starting head state of the machine. We can assume without loss of generality that the Turing machine never transitions back to the q_0 state, and never has a transition that would move it left from the first location on the tape. Given that we have one row of this form, the next row up must have the same number of pairs of variety 2 and variety 3 tiles, and furthermore, each pair must be shifted one position left or right, with the Turing machine performing an allowed transition, as in the example. If one row has exactly one variety 2 tile $[a,q_0,l]$ in the leftmost position, then every row above it in a valid tiling will also have exactly one variety 2 tile, and the jth row up will correctly represent the state of the Turing machine tape and head after j steps.

For our particular tiling, we will need additional rules for some pairs of boundary tiles and interior tiles. These additional rules will be different for layers 1 and 2, and some of them will couple the two layers. In addition, we will need to slightly modify some of the horizontal tiling rules. The new and modified rules are summarized in Table 4. Also, for layer 1, recall that we reverse "above" and "below" for the regular Turing machine simulation rules (Table 3), so that time goes downwards instead of upwards.

We would like to start out the Turing machine M_{BC} with $[\#, q_0, l]$ in the leftmost location followed by $[\#]$ tiles. This is ensured by the additional rules for layer 1: The top interior row must consist of only these two types of tiles, and the leftmost location cannot be $[\#]$. Since there cannot be any variety 3 tiles, there can only be one $[\#, q_0, l]$ in the leftmost location. We can assume without loss of generality that the Turing machine overwrites the leftmost # on the tape and never writes a # there again.

The rest of the layer 1 rules just enforce the rules for the Turing machine M_{BC}.

Now in order to copy the output from M_{BC} to the input tape for M, we restrict the kinds of tiles that can go above tiles. All the alphabet characters in the bottom interior row of layer 2 must match the alphabet characters for layer 1. That is, the output of M_{BC} is copied onto the input of M.

We also want to ensure that the starting configuration of M has only one head in the leftmost location. To accomplish this, we forbid a to go next to an $[a]$ tile for $a \in \Sigma_{M_{BC}}$, so the leftmost tile in the 2nd row from the bottom of layer 2 must be $[a, q_0, l]$. Since there can be no variety 3 tiles in the 2nd row, that must be the only variety 2 tile in the row. A little care must be taken to overwrite the leftmost input tape character with something that is not in the alphabet of M_{BC}. This is because we have forbidden having an $[a]$ tile to the right of a for any $a \in \Sigma_{M_{BC}}$, and this prohibition applies to *all* rows. The information encoded in the left-most tape symbol can be retained by having a new a' symbol in Σ_M for every $a \in \Sigma_{M_{BC}}$.

Finally, the only variety 2 tiles on layer 2 which we allow below a tile must be of the form $[a, q_A, r/l]$, 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. $\square$

4 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. Also, we prove that in one dimension, TILING and all the above variants are easy.

All of the hardness results are proven by reducing from TILING, perhaps with a restriction (usually straightforward) on the value of N. For instance, when discussing WEIGHTED TILING with periodic boundary conditions, we require N to be odd. TILING with odd N is clearly still NEXP-complete, as we can either use a universal Turing Machine M that performs some processing on the input x, or simply a different counting Turing Machine M_{BC} that counts more slowly, effectively ignoring the least significant bit of N. The other restrictions on N we use similarly result in hard subclasses of TILING.

To prove the variants hard, we have a main layer, which duplicates the tiling rules given to us, with perhaps some small variations (e. g., adding extra tiles). We also add additional layers with tiling rules that force a structure that duplicates the conditions of the standard TILING problem. The details of the additional layer differ with each construction, and are provided in the discussion below.

4.1 Boundary conditions

Our choice of fixing the tiles in all four corners of the square is unusual. See, for instance, Papadimitriou [29], 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:

Observation 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 observation follows immediately from the fact 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 [4]. 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. Note that this observation also holds for open boundary conditions. We only get hardness results in this case when we move to the weighted version of the problem in Section 4.2.

On the other hand, if we fix boundary conditions in two corners, that is already enough for hardness. There are two cases to consider: when the two corners are adjacent and when they are opposite. The techniques used for showing these two cases are hard are similar to those used in some other variants, so we omit the details. When the two corners are opposite, we can create a boundary much like that of Figure 1, but with two new unique corner tiles. When the two corners are adjacent (assume without loss of generality that these are the top and bottom left corners), we can modify the tiling rules given in Section 3 in order to avoid the need for the right boundary. We only need special tiling conditions at the right boundary to prevent a new TM head from appearing there. If instead, we insist that only [#] can appear to the right of [#], it is also impossible to create a new TM head on the right end.

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 4.1. *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. The randomized reduction has zero error and runs in expected polynomial time.*

The reduction requires that we find prime N, so we are not able to do the reduction in polynomial time. 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 poly(N), then, as a consequence of Theorem 4.1, $\mathsf{EXP} = \mathsf{NEXP}$.

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

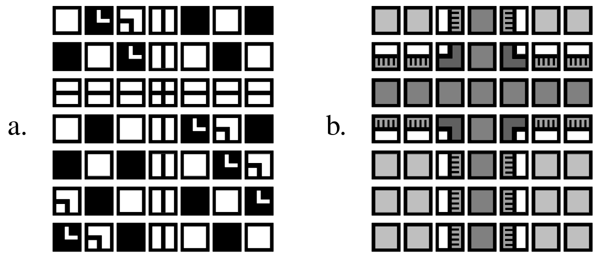


Figure 2: A possible arrangement of layers 1 (a.) and 2 (b.) for PERIODIC TILING.

types of tile: , , , , , , and . The new layer 2 has 10 types of tile: , , , , , , , , , and .

The , , and tiles will create the vertical and horizontal borders. and are used to make sure that there is at least one of each kind of border, and the and 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. The structure of tiling that we would like to achieve is shown in Figure 2.

Layer 1 The tiling rules for layer 1 are summarized in Table 5.

The rules for , , and imply that if we have a anywhere in a column, that column must only contain and tiles, and if we have a anywhere in a row, that row can only contain and tiles. Furthermore, a tile must be surrounded by tiles above and below and tiles to the left and right. Thus, layer 1 contains some number of vertical and horizontal lines composed of and tiles intersecting at 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 or alternating with or tiles. We ensure this by forbidding and from being adjacent to themselves or each other in any direction, and forbidding and 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 tile anywhere, we want to be forced to have a diagonal line of tiles continuing to the upper left and lower right, with a diagonal line of 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 tile must have both below it and to its left. Above a tile we can have only or , and to the right of a tile, we must have either or .

Layer 2 The tiling rules for layer 2 are summarized in Table 6.

We wish to mark the top side of the rectangles delineated by the layer 1 and tiles. will mark the right side, the left side, and the bottom side, with , , , and marking the upper left, upper right, lower left, and lower right corners, respectively. will be inside the rectangle and will go over the layer 1 and tiles.

Tile on left \ Tile on right	⊟	◫	⊞	□	◳	■	◘
⊟	Y	N	Y	N	N	N	N
◫	N	N	N	Y	N	Y	Y
⊞	Y	N	N	N	N	N	N
□	N	Y	N	N	N	Y	Y
◳	N	Y	N	N	N	Y	N
■	N	Y	N	Y	N	N	N
◘	N	Y	N	N	Y	N	N

Tile on bottom \ Tile on top	⊟	◫	⊞	□	◳	■	◘
⊟	N	N	N	Y	N	Y	Y
◫	N	Y	Y	N	N	N	N
⊞	N	Y	N	N	N	N	N
□	Y	N	N	N	N	Y	Y
◳	Y	N	N	N	N	Y	N
■	Y	N	N	Y	N	N	N
◘	Y	N	N	N	Y	N	N

Table 5: The tiling rules for layer 1 for PERIODIC TILING.

Looking at the horizontal tiling rules, we see that if we have a anywhere, there must be a horizontal line of that either goes all the way around the grid (due to the periodic boundary conditions) or ends at on the left and on the right. If it ends, then, because of the vertical tiling rules, below the tile is a vertical line of tiles which end at a tile, and below the tile is a vertical line of tiles which end at a tile. The and tiles must be in the same row, and between them is a line of tiles, forming a closed rectangle. 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.

The remaining rules enforce that we have on the outside of the rectangles and 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. These rules imply that inside the rectangle delineated by the , , , and tiles are only tiles, and immediately outside it are tiles. Each rectangle is outlined by , , , and tiles on the sides, with , , , and on the corners, and is full of inside. A vertical stripe has on its left and on its right, and a horizontal stripe has above it and below it. Both vertical and horizontal stripes have only inside them. The rectangles and stripes cannot be adjacent and are separated by tiles.

Interactions between layers 1 and 2 The layer 1 and layer 2 tiles must pair up as indicated in Table 7.

These conditions tell us that the horizontal and vertical lines on layer 1 formed by ⊟, ◫, and ⊞ must match exactly the locations of the tiles on layer 2. This implies 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 ◘ in the

	Tile on right									
	Y	N	N	N	N	Y	N	N	N	N
	N	Y	N	N	N	N	N	Y	N	N
	N	N	N	N	N	N	N	N	N	Y
Tile	N	N	N	N	N	N	N	N	Y	N
on	Y	N	N	N	N	N	N	N	N	N
left	N	N	N	N	N	N	N	N	N	Y
	N	Y	N	N	N	N	N	N	N	N
	N	N	N	N	N	N	N	N	N	Y
	N	N	Y	N	N	N	N	N	Y	N
	N	N	N	Y	Y	N	Y	N	N	Y

	Tile on top									
	N	N	N	N	N	N	N	N	N	Y
	N	N	N	N	N	N	N	N	Y	N
	N	N	Y	N	N	Y	N	N	N	N
Tile	N	N	N	Y	Y	N	N	N	N	N
on	N	N	N	N	N	N	N	N	N	Y
bottom	N	N	N	N	N	N	N	N	N	Y
	N	N	N	Y	N	N	N	N	N	N
	N	N	Y	N	N	N	N	N	N	N
	Y	N	N	N	N	N	N	N	Y	N
	N	Y	N	N	N	N	Y	Y	N	Y

Table 6: The tiling rules for layer 2 for PERIODIC TILING.

	Layer 2 tile									
	N	N	N	N	N	N	N	N	N	Y
	N	N	N	N	N	N	N	N	N	Y
Layer 1	N	N	N	N	N	N	N	N	N	Y
tile	Y	Y	Y	Y	N	Y	Y	N	Y	N
	Y	Y	Y	Y	N	Y	Y	N	Y	N
	Y	Y	Y	Y	N	Y	Y	N	Y	N
	N	N	N	N	Y	N	N	Y	Y	N

Table 7: The allowed pairs of layer 1 and layer 2 tiles for PERIODIC TILING.

upper left corner, which starts a diagonal line of tiles extending towards to the bottom right. Since it must end at the border of the rectangle, but a tile cannot be in the same spot as a layer 2 or 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 2 for an example of an allowed 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 3. The layer 2 , , , and 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 than x ($\log N = \mathrm{poly}(\log x)$), and it must be possible for the universal TM implemented by the data layers to deduce x in a reasonable time. We will show a method of finding a prime N such that the $1/3$ most significant bits represent x. 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). We want to show that there exists a prime number in the range $[N_0(x), N_0(x+1))$. In fact, we will show that there is a prime in a somewhat narrower range so that it can be found by exhaustive search in polynomial space. Furthermore, the primes in this range are sufficiently plentiful that the expected time to find a prime number by random selection will be polynomial.

It is known that there exists a constant $\theta < 1$ such that

$$\lim_{y \to \infty} [\pi(y + y^\theta) - \pi(y)] = \frac{y^\theta}{\log y}, \tag{4.1}$$

where $\pi(y)$ is the number of primes less than or equal to y [14]. (For our purposes, it is sufficient to take $\theta = 2/3$, but smaller θ 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.[2] Then

$$N_0(x)^\theta = x^\theta 2^{\theta n_0(x)} \leq 2^{\theta(\lceil \log x \rceil + n_0(x))} \leq 2^{n_0(x)}. \tag{4.2}$$

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 (4.1), 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 $\mathrm{poly}(\log x)$ time. Thus, we get a randomized reduction to PERIODIC TILING which runs in expected time $\mathrm{poly}(\log x)$. □

[2] 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.

	Tile on right				
Tile on left	+4	+4	+4	+4	-1
	+4	+4	+4	+4	+2
	+4	+4	+4	+4	-1
	+4	+4	+4	+4	+2
	+2	-1	+2	-1	0

	Tile on top				
Tile on bottom	+4	+4	+4	+4	+2
	+4	+4	+4	+4	+2
	+4	+4	+4	+4	0
	+4	+4	+4	+4	0
	0	0	+2	+2	0

Table 8: The tiling weights for WEIGHTED TILING with open boundary conditions.

4.2 Weighted constraints

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

Definition 4.2. WEIGHTED TILING
Problem Parameters: A set of tiles $T = \{t_1, \dots, t_m\}$. A set of horizontal weights $w_H : T \times T \to \mathbb{Z}$, such that if t_i is placed to the left of t_j, there is a contribution of $w_H(t_i, t_j)$ to the total cost of the tiling. A set of vertical weights $w_V : T \times T \to \mathbb{Z}$, such that if t_i is placed below t_j, there is a contribution of $w_V(t_i, t_j)$ to the total cost of the tiling. A polynomial p. Boundary conditions (a tile to be placed at all four corners, open boundary conditions, or periodic boundary conditions).
Problem Input: Positive integer N, specified in binary.
Output: Determine whether there is a tiling of an $N \times N$ grid such that the total cost is at most $p(N)$.

Note that we can shift all the weights by a constant r:

$$w'_H(t_i, t_j) = w_H(t_i, t_j) - r \quad \text{and} \quad w'_V(t_i, t_j) = w_V(t_i, t_j) - r\,.$$

This has the effect of shifting the total cost by $2rN(N-1)$. Therefore, we may assume without loss of generality that $p(N) = o(N^2)$.

For simplicity, we have considered only the case where p is a polynomial, in this case a linear or constant one. This is sufficient for our constructions, but there may also exist interesting weighted tiling systems where the minimal total cost scales as some other function, such as $\Theta(\sqrt{N})$.

Theorem 4.3. *WEIGHTED TILING is* NEXP*-complete, with any of the three choices of boundary conditions.*

Proof. The case of the four-corners boundary condition follows immediately from the result for unweighted TILING.

WEIGHTED TILING with open boundary conditions It is sufficient to take all the weights to be 0, $+2$, $+4$, or -1, and $p(N) = -4$. 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 consisting of five special types of tile, , , , , and . The weights are summarized in Table 8.

Suppose we have any configuration which contains the tile , and suppose we replace that tile by , leaving the rest of the configuration the same. If there is any tile to the left of the replaced tile, the cost of

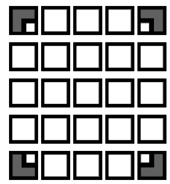


Figure 3: The preferred arrangement of the new layer in WEIGHTED TILING with open boundary conditions.

		Tile on right				
		□	■	⊟	◫	⊞
	□	+3	0	+3	0	+3
Tile	■	0	+3	+3	0	+3
on	⊟	+3	+3	0	+3	+1
left	◫	0	0	+3	+3	+3
	⊞	+3	+3	+1	+3	+3

		Tile on top				
		□	■	⊟	◫	⊞
	□	+3	0	0	+3	+3
Tile	■	0	+3	0	+3	+3
on	⊟	0	0	+3	+3	+3
bottom	◫	+3	+3	+3	0	0
	⊞	+3	+3	+3	0	+3

Table 9: The tiling weights for WEIGHTED TILING with periodic boundary conditions.

that edge decreases by at least 2. Similarly, if there is any tile above the replaced tile, the cost of the above edge decreases by at least 2. The only way for the overall cost of the tiling to increase is if the replaced ◰ tile was located in the upper left corner. Similarly, if we replace ◳ by □, the cost decreases unless possibly the ◳ tile is located in the upper right corner. Similarly, we can more cheaply replace ◱ and ◲ by □ tiles unless the ◱ and ◲ tiles are located in the lower left and lower right corners, respectively. Thus, the cheapest tilings have □ tiles everywhere but the corners, and it is easy to verify that the optimal tiling of the new layer has total cost -4, with ◳, ◰, ◱, and ◲ in the proper corners, and □ elsewhere. Figure 3 shows the optimal configuration of the new layer.

Finally, we insist that ◳, ◰, ◱, or ◲ in the special layer must correspond to the usual corner tile t_1 in the main layer. Then the optimal tiling overall has the main layer 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 4.1, 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. We will use the weights 0, $+1$, and $+3$, and set $p(N) = +2$. Thus, to have a good enough tiling, we can have two pairings with weight $+1$, and all the others must have weight 0. The weights we use are summarized in Table 9.

In order to avoid any pairing with cost $+3$, we cannot have □ and ■ next to each other. That is, any region with just these two tile types must be in a checkerboard pattern. Of course, when N is odd, we cannot fill the whole grid that way. Indeed, in each row and column, there must be at least one ⊟, ◫, or ⊞ tile.

If we have a ⊞ tile anywhere, it must have a ⊟ tile adjacent to it to the left and right, and a ◫ tile adjacent to it above and below. Whenever we have a ◫ tile, it must form a vertical line with only ◫ and ⊞ tiles wrapping all the way around the torus. Similarly, if we have a ⊟ tile, there must be a horizontal line with only ⊟ and ⊞ tiles. Furthermore, if there is a single ⊞ tile in the horizontal line, the overall cost of the edges within the line is $+2$, and if there is more than one ⊞ tile, the cost is larger.

Since ⊟ tiles cannot be adjacent vertically, and ◫ tiles cannot be adjacent horizontally, nor can either be adjacent to the other in any direction, the only way to avoid a full row or column with only ■ and □ tiles is to have at least one horizontal line and at least one vertical line. Whenever a horizontal line and vertical line intersect, there must be a ⊞ at the crossing. Each crossing has a cost of $+2$, so to minimize the total cost, we can have only one crossing, and thus just one horizontal line of ⊟'s and one vertical line of ◫'s. 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 ⊟ and ◫ tiles. The arrangement is much like that of Figure 2a, but with ◩ replaced by ■ and ◰ replaced by □. □

4.3 One-dimensional tiling

In one dimension, the tiling problem becomes the following:

Definition 4.4. 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: Positive 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 4.2.

Theorem 4.5. 1*-DIM TILING and WEIGHTED* 1*-DIM TILING are in* P.

Proof. **Unweighted case** We will create a directed graph with m nodes in which the ith node corresponds to the ith tile type t_i, and there is an edge between i and j iff $(t_i,t_j)\in H$. Let M be the matrix representation of this graph. We wish to determine whether there is a path of length exactly N starting and ending at t_1. This is the same as determining whether the matrix M^{N-1} has a non-zero entry in the diagonal entry corresponding to t_1. In fact, as long as M^{N-1} can be computed efficiently (in $O(\log N)$ time), one can generalize this to the case where there are different boundary conditions on the two ends of the line. If t_i must be on the left and t_j must be on the right ends of the chain, we need to determine if the entry corresponding to (t_i,t_j) in M^{N-1} is non-zero.

Since the matrix M is of constant size, we can compute M^{N-1} as follows: For each $2^i\leq N-1$, compute M^{2^i} by squaring the matrix M i times. Then compute M^{N-1} by multiplying the M^{2^i} for each i such that the binary representation of $N-1$ has a 1 in the i^{th} location.

Weighted case For the weighted case, we construct a new matrix M_w such that the entry corresponding to (t_i,t_j) is $w(t_i,t_j)$, the cost of putting a tile of type t_i to the left of a tile of type t_j. However, the matrix should be considered as acting over a non-standard algebra, one for which the multiplication operation is replaced by addition and the addition operation is replaced by the minimum operator. One can check that these operations give a commutative algebra (the distributive law is the main thing to check) and that matrix multiplication over this algebra is associative.

Now, we claim that the value of entry (t_i, t_j) in $(M_w)^{N-1}$ is the weight of the minimum weight tiling with a t_i tile in the leftmost location and a tile of type t_j in the rightmost location. This follows by induction, noting that the (t_i, t_j) entry in $(M_w)^N$ is

$$[(M_w)^N]_{(t_i,t_j)} = \min_k \left([(M_w)^{N-1}]_{(t_i,t_k)} + (M_w)_{(t_k,t_j)} \right), \tag{4.3}$$

which is the minimum, over k, of the minimum weight of a tiling of size $N-1$ starting with t_i and ending with t_k, plus the cost to have the last two tiles be t_k followed by t_j.

We can compute $(M_w)^{N-1}$ using the same method as for the unweighted case. Therefore the problem is in P. □

With a more careful analysis, it transpires that both the weighted and unweighted 1-DIM TILING problems are in fact in NC^1. For instance, for the weighted case the corresponding language is accepted by a non-deterministic pushdown automata when N is given in unary. This implies by *Parikh's Theorem* [31] that the set is semi-linear which in turn means that the corresponding language is semi-linear and hence in NC^1 as well.

4.4 Additional symmetry

We consider two additional kinds of symmetry. If we have *reflection symmetry*, then $(t_i, t_j) \in H$ implies $(t_j, t_i) \in H$ as well, and $(t_i, t_j) \in V$ implies $(t_i, t_j) \in V$ 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 ) that looks different when it is turned upside down, then when we reflect vertically, we also could reflect the tile, producing a new tile $R(t_i)$ ( 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 4.6. *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 $N_e = N_o = 1$.*

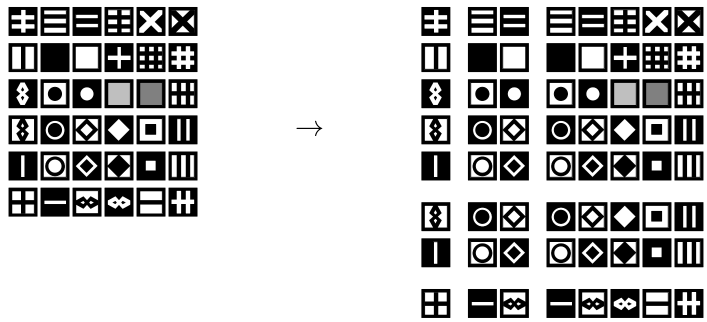


Figure 4: Extending a tiling of a 6×6 grid to 8×8 when the tiling conditions have a reflection symmetry. The spaces on the right mark the repeated rows and columns.

- *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.

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 2×2 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 4.)

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 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 2×2 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 5.) This gives us a tiling of the 3×3 grid with all four corners correctly tiled. Conversely, if there is no 2×2 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 $N_o = N_e = \infty$.

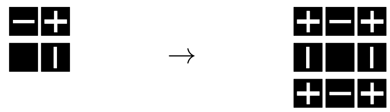


Figure 5: Extending the tiling of a corner to a tiling of a 3×3 grid with fixed corner boundary conditions.

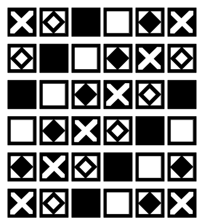


Figure 6: A tiling of a 6×6 square with rotation symmetry.

Open boundary conditions In the case of completely open boundary conditions, we need only check a 2×2 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 5 to get a tiling of the 3×3 grid as well, and extend as in Figure 4 to get a tiling of any size grid.

Periodic boundary conditions When we have periodic boundary conditions, a valid tiling of a 2×2 grid with open boundary conditions gives us a valid periodic tiling of a 2×2 grid as well, so we can extend it to all even N. However, if we try to apply the strategy of Figure 5 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 2×2 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 6. 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. $\square$

The weighted cases with reflection or rotation symmetry are more difficult, so we treat them in separate subsections.

4.5 Weighted tiling with reflection symmetry

In this subsection, we consider WEIGHTED TILING with reflection symmetry. Reflection symmetry means that the functions w_H and w_V are independent of the order of their arguments: $w_H(t_i,t_j) = w_H(t_j,t_i)$ and $w_V(t_i,t_j) = w_V(t_j,t_i)$.

4.5.1 Open or four corners boundary conditions

Theorem 4.7. *WEIGHTED TILING with reflection symmetry is* NEXP*-complete with either open boundary conditions or boundary conditions fixed at the corners.*

It is interesting to note that the total cost $p(N)$ of the satisfying tilings that appear in our proof is linear in N. We have not been able to prove a result for these boundary conditions when the cost function is a constant. (Recall that we can always shift the costs so that $p(N) = o(N^2)$.)

Proof. We will describe the costs for open boundary conditions, but they will imply fixed corner tiles, so the four-corners boundary conditions can use the same costs. When we list costs, we will talk of pairings being "forbidden." Of course, in WEIGHTED TILING, no pair of adjacent tiles is completely forbidden, but we will assign a large cost (say $+30$) to "forbidden" pairings, and will show that a low-cost tiling can never include a forbidden pair.

The proof makes use of three extra layers beyond the main layers used to prove the basic NEXP-completeness of TILING, allowing us to reduce standard TILING to WEIGHTED TILING with reflection symmetry. The rules from Section 3 are not symmetric; our goal, therefore, is to define extra layers which use symmetric weight functions to produce an asymmetric state which can be used to orient the rules in the main layers. By cross-referencing the asymmetric state from the new layers with the state of the main layers, symmetric weights can be used to simulate asymmetric rules.

Layer 1, the first of the new layers, will break the reflection symmetry in both the horizontal and vertical directions, but the broken symmetry will only be visible at certain locations in the grid. That is, there is not a unique optimal tiling of Layer 1. Any optimal tiling can be globally reflected in the horizontal and/or vertical directions to get another optimal tiling. We will choose rules such that one of these reflections, applied to an optimal tiling, always produces a distinct tiling. There will, in fact, be exactly 4 optimal tilings of Layer 1, related to each other by reflections. (There is actually another class of optimal tilings of Layer 1 when it is considered by itself, but that class will be eliminated by the rules for Layers 2 and 3.) Since the reflected tilings are different from each other, there are certain locations we can look at to determine which orientation of the four we actually have. By choosing one of the four orientations to be a canonical reference point, we can globally define the four directions "left," "right," "up," and "down." We would like to use this information to implement direction-dependent rules for the main layers even though the underlying rules remain reflection-invariant. To do this, we need to be able to look at an adjacent pair of tiles in Layer 1 and determine a direction from that. Since the rules are translation-invariant, we cannot make reference to the location of the tiles, only that they are adjacent in either the horizontal or vertical direction. Nevertheless, for an optimal tiling of Layer 1, it is possible in some locations in the grid to determine directions purely from the Layer 1 tiles. However, it is not possible everywhere.

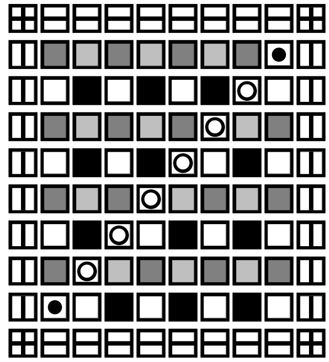


Figure 7: A tiling for Layer 1 of a 10×10 grid for the WEIGHTED TILING problem with reflection symmetry. The reflection symmetry is broken in the immediate vicinity of the central diagonal.

Therefore, we use Layer 2 to extend the broken horizontal reflection symmetry, so there is a locally visible distinction between "left" and "right" at all locations in the grid. Layer 3 similarly extends the broken vertical reflection symmetry, allowing us to define "up" and "down" at all locations in the grid. Most of the work, and all of the non-trivial weights, go into Layer 1. Layers 2 and 3 only have weights 0 and $+30$ (for forbidden pairings).

We will restrict N to be even and 1 mod 3. That is, $N \equiv 4 \bmod 6$. We will set the allowed cost to be $p(N) = 76 - 16N$, and assume N is large.

Layer 1 Layer 1 will have 9 types of tile: , , , , , , , , and . , , and will form the outer border of the grid in an optimal tiling. The interior of the grid will mostly consist of , , , and tiles. By and large, these four tiles will alternate and or and horizontally, and and or and vertically. However, there will also be a diagonal stripe of tiles reaching from corner to corner, with tiles at the end. At the left and right ends of the interior, adjacent to the tiles, we only permit and tiles (alternating vertically) or tiles. Because N is even, the tiles to the left and right of the and tiles will be different in a consistent way, allowing us to distinguish left and right near the central diagonal. Similarly, the tiles above and below the and tiles are different, allowing us to distinguish up and down. See Figure 7 for an example of this tiling.

To achieve this, we use the rules given in Table 10.

Claim 4.8. *With these rules, for large N, the cost of an optimal tiling is* $76 - 16N$.

Our approach will be to try to optimize the cost of tiling each row, and then to combine the optimal rows to get the overall tiling. Since we care about the cost of vertically adjacent tiles as well as horizontally adjacent tiles, we will actually optimize adjacent *pairs* of rows. If we add up the cost of pairs of rows, however, we end up double counting the horizontal costs. The solution is to optimize pairs of rows with respect to a modified cost function, given in equation (4.8) below. This necessitates treating the top two rows and the bottom two rows separately, since the top and bottom rows are not double counted. A second complication arises from the possibility that the globally optimal tiling will not optimize all adjacent pairs of rows. In order to rule out this, we also consider some sub-optimal tilings of pairs of rows.

Horizontal tiling rules

	30	30	30	30	6	6	30	30	7
	30	-11	0	30	30	30	30	30	30
	30	0	30	30	30	30	30	30	30
	30	30	30	30	0	30	30	1	30
	6	30	30	0	30	30	30	1	1
	6	30	30	30	30	30	0	1	1
	30	30	30	30	30	0	30	1	30
	30	30	30	1	1	1	1	30	30
	7	30	30	30	1	1	30	30	30

Vertical tiling rules

	-11	30	0	30	30	30	30	30	30
	30	30	30	6	6	6	6	30	7
	0	30	30	30	30	30	30	30	30
	30	6	30	30	30	30	0	1	30
	30	6	30	30	30	0	30	1	1
	30	6	30	30	0	30	30	1	1
	30	6	30	0	30	30	30	1	30
	30	30	30	1	1	1	1	30	30
	30	7	30	30	1	1	30	30	30

Table 10: The tiling weights for layer 1 for WEIGHTED TILING with reflection symmetry. Since there is reflection symmetry, the horizontal and vertical tiling weight matrices are symmetric.

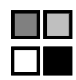
To show that the desired tiling is indeed optimal, let T be some tiling, with T_{ab} the identity of the tile in location (a,b) in the grid. Let $w(T)$ be the total cost of T and let $w(S_{ab})$ be the total cost of the 2×2 square in location (a,b): i. e.,

$$w(S_{ab}) = w_H(T_{ab},T_{(a+1)b}) + w_V(T_{ab},T_{a(b+1)}) + w_V(T_{(a+1)b},T_{(a+1)(b+1)}) + w_H(T_{a(b+1)},T_{(a+1)(b+1)})\,. \tag{4.4}$$

Finally, let $w(R_b) = \sum_a w_H(T_{ab},T_{(a+1)b})$ be the internal cost of row b and $w(C_a) = \sum_b w_V(T_{ab},T_{a(b+1)})$ be the internal cost of column a. Then

$$2w(T) = \sum_{a,b=1}^{N-1} w(S_{ab}) + w(R_1) + w(R_N) + w(C_1) + w(C_N)\,. \tag{4.5}$$

The only negative weights are the horizontal edges and the vertical edges between two tiles. There are no possible 2×2 squares with negative cost, and the only 2×2 squares with total cost 0 are



(4.6)

and its three reflections. Therefore, to get a minimal cost tiling, we should have as many 2×2 squares as possible be of that form, and have preferentially and around the edges of the grid.

We can break the whole cost down into pairs of rows. Let us calculate the minimal value of

$$w'(R_b,R_{b+1}) \equiv w(R_b) + w(R_{b+1}) + 2\sum_{a=1}^{N} w_V(T_{ab},T_{a(b+1)}) \tag{4.7}$$

$$= \sum_{a=1}^{N-1} w(S_{ab}) + w_V(T_{1b},T_{1(b+1)}) + w_V(T_{Nb},T_{N(b+1)}) \tag{4.8}$$

for a particular assignment of tiles to a pair of rows. We must decide a collection of N vertical pairs of tiles to achieve the minimum cost. Since

$$2w(T) = \sum_b w'(R_b,R_{b+1}) + w(R_1) + w(R_N)\,, \tag{4.9}$$

the minimal value of w' for pairs of rows is a good guide to the minimal achievable total cost for the whole grid. By equation (4.8), if every square in the pair of rows had cost 0, and the first and last pairs had the minimal edge cost -11, we would have $w'(R_b,R_{b+1}) = -22$, but we cannot quite achieve that:

Lemma 4.9. *When $N > 4$ is even, the minimum value of $w'(R_b,R_{b+1})$ is -12.*

Any pair of rows R_b, R_{b+1} with $w'(R_b,R_{b+1}) = -12$ has the following structure: Each row starts and ends with tiles, has exactly one or tile, and to the left and right of the or, alternates either and or and. The and/or tiles in the two rows are diagonally adjacent to each other. Furthermore, if one row has alternating and tiles to the right (left) of the or tile, the other row has alternating and tiles to the right (left) of the or tile, as required by the allowed pairs of adjacent tiles.

Beyond this structure, there are four classes of solutions:

a. One row contains a tile adjacent to one of the tiles. The other tile in the column with the tile is the same (or) as the tile in the column with the tile in the other row.

b. One row contains a tile adjacent to one of the tiles. The other tile in the column with the tile is different than the tile in the column with the tile in the other row. In this case, one row contains no or tiles and the other row contains no or tiles.

c. Both rows have tiles. The two tiles in the same columns as the tiles are the same. In this case, each row contains at least one tile from each pair (,) and (,).

d. Both rows have tiles. The two tiles in the same columns as the tiles are different. In this case, each row contains tiles from only one of the pairs (,) and (,).

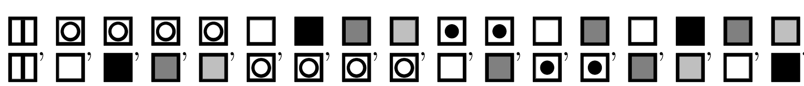
If one side (left or right) is forbidden to have a for one or both rows, the minimal value achievable is -10. *This can be achieved only through a pair of rows which uses at the other end of both rows, and alternates and elsewhere in one row, and and elsewhere in the other row.*

If is forbidden for both ends of one or both rows, the minimal value achievable is 0. *This can only be achieved by having one row alternating and , and the other row alternating and .*

The four classes of minimal-cost solution given by the lemma depend on two factors. First, do we have only tiles, or one tile and one tile? Classes a and b have , whereas classes c and d do not. Second, do we have the same alternating pair (,) or (,) on both sides of the tile(s), or do we have different alternating pairs on the left and right? Classes a and c have different alternating pairs, while classes b and d have the same pair on each side of the tile in a particular row. In our solution, we are primarily interested in classes b and d because when N is even, they have an asymmetry between left and right in the vicinity of the tiles. However, at this stage we cannot rule out classes a and c.

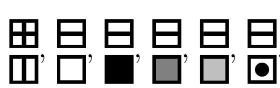
Proof of lemma. The minimal cost of a pair of rows could be calculated much as in Section 4.3. We merge each vertically adjacent pair of tiles into a single node of the graph. The difference from Section 4.3 is that to minimize $w'(R_b, R_{b+1})$, we must now assign a cost to visiting a node equal to $2w_V(T_{ab}, T_{a(b+1)})$ in addition to the cost of each edge (which is equal to $w_H(T_{ab}, T_{(a+1)b}) + w_H(T_{a(b+1)}, T_{(a+1)(b+1)})$). We won't use the algorithm from Section 4.3 because we also need more detailed information about what the cheapest paths are.

Let us first consider paths containing no forbidden pairings. Consider the graph whose nodes are allowed vertical pairs and edges are allowed horizontal 2×2 squares. The nodes of the graph are weighted by twice the cost of the horizontal edge in the pair, and the edges of the graph are weighted by the sum of the cost of the two vertical edges in the square. The graph breaks down into three connected components. The first component contains the pairs



(4.10)

The second component contains the pairs



(4.11)

The third component is just the vertical mirror image of the second component. In the second and third components, there are no paths with total cost w' less than or equal to 0: The vertical pairs which include have cost per node of $+12$ or $+14$, and the minimal cost of an edge is -11. The vertical pair which includes only has node cost 0, but all the edges leaving it also have positive cost ($+6$ or $+7$). We will be able to achieve path costs less than 0 using the first component, so the second and third components are only useful at the top and bottom edges of the $N \times N$ grid.

Let us consider in more detail the first component, pictured in Figure 8. First, the costs of the nodes: The pair with two tiles has node cost -22. The 12 nodes which involve a or have node cost $+2$, and the remaining 4 nodes have cost 0. However, the edges from the node with two tiles have large positive cost: $+12$ to connect to the nodes involving both and , and $+13$ to connect to the 4 nodes involving . (It is not adjacent to the other nodes.) The pairs involving or have edge costs $+1$ (to connect to a node which does not involve , , or), $+2$ (to connect to another node with a or), or $+13$ (to connect to the two- node, which is only possible for the four nodes). The remaining four nodes involve either the pair (,) or the pair (,). Besides the connections listed above, they also connect to one of the other four nodes in this class with an edge of cost 0.

To determine the structure and cost of paths, we break them up into simple cycles and simple paths. A *simple path* is a path containing no repeated nodes, except perhaps the first and final ones. The cost of a simple path will be the sum of weights of the edges and nodes in the path. A *simple cycle* is a cycle for which the only repeated node is the initial (and final) node. The cost of a simple cycle is the sum of the weights of all the edges and all nodes, except that the repeated node is only counted once.

Note that given any path, we can extend it by inserting a simple cycle. Just choose any node in the original path, and replace that node with the simple cycle. Furthermore, the cost of the extended path (the sum of weights for all nodes and edges) is equal to the cost of the original path plus the cost of the simple cycle. This is why we counted the repeated node only once in the cost of a simple cycle—because we have to remove the node from the original path before inserting the simple cycle.

We can also do this process in reverse, taking a path which is not a simple path and cutting out a simple cycle: Find a repeated node in the path. Then the path from one occurrence of that node to the next one is a simple cycle. Replace it with a single copy of the repeated node, giving us a shorter path with the same start and end points. Re-inserting, as above, the same simple cycle at the same node restores the original path. Therefore the cost of the longer path is equal to the cost of the shorter path plus the cost of the simple cycle. By repeatedly cutting out simple cycles until we are left with a simple path, we can break the cost of any path down into the cost of a simple path plus the total cost of the simple cycles composing it. Note that the process might not be unique (there may be more than one way to choose simple cycles to remove), but that doesn't affect the analysis of the cost.

Based on this, let us first consider the cost of simple cycles. We are particularly interested in minimum-cost simple cycles of various lengths. For a simple cycle, we count the node cost for the beginning/ending node only once. It is thus easy to see that we cannot achieve a negative cost simple cycle—the benefit (negative cost) of having a (,) node is outweighed by the high positive cost of the edges in and out of that node. Therefore, the minimum cost of a cycle is 0, and this is easy to achieve for a cycle of length 2, using a (,) node and the matching (,) node.

Thus, if our path contains any (,) or (,) node, we can extend it to a path whose length has the same cost and the same parity (even or odd) by inserting copies of an appropriate cost 0 cycle. In

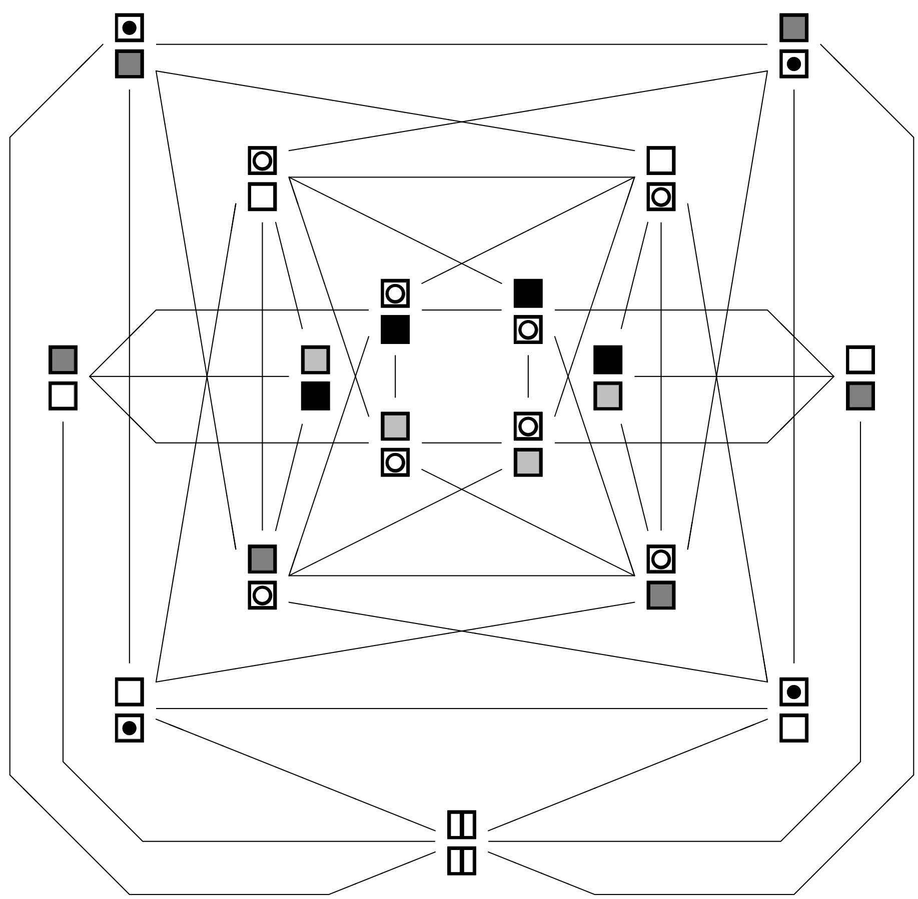


Figure 8: The first component of the graph of vertical pairs of tiles.

order to cover paths whose length might have a different parity than the simple path we started with, we will also need to insert a simple cycle of odd length. There are no other cost 0 simple cycles, so the minimum-cost odd length cycle will have positive cost, and we will only want to use at most one of them. It is not possible to build an odd cycle of allowed transitions using just (◫, ◫), (■, ▒), and (□, ▓) nodes, so we will need to include at least one node with a ⊡ or ⧇ tile. Since there is a cost associated to the nodes involving ⊡ or ⧇ tiles, we would like to minimize the number of such nodes we use. It is straightforward to see there are no odd cycles with only one ⊡ or ⧇ node, so we will want cycles which use two of these nodes.

Let us first consider odd simple cycles that contain a (■, ▒) or (□, ▓) node. In this case, we can achieve a cycle with total cost $+8$ and length 3 by starting with a (■, ▒) or (□, ▓) node, followed by two nodes involving ⧇, and then back to the original node. There are a number of allowed paths of this form, such as

□ ■ ⧇ □
▓ ⧇ ▒ ▓ .   (4.12)

There are no other allowed length 3 cycles starting and ending on a (■, ▒) or (□, ▓) node. There are longer simple cycles, but they also have larger cost.

The other minimal cost odd simple cycle starts on a (◫, ◫) node, connecting from there to any of the four ⊡ nodes, and from there to another ⊡ node (with the ⊡ in the opposite position), then back to the (◫, ◫) node. The total cost is $+10$. This type of length 3 cycle costs more than the previously discussed class of length 3 cycles, so would only be useful if we wanted to extend a path that contained no (■, ▒) or (□, ▓) nodes. This will not be needed, so we can ignore this simple cycle.

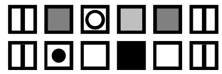
The next step is to consider minimal cost simple paths. Our goal is to achieve an overall negative cost, and it is clear the only way to achieve that is through the use of (◫, ◫) nodes. In particular, the only possible locations for the (◫, ◫) nodes are the beginning and the ending of our path, since those are the only locations where the cost of edges in and out of the node does not overcome the negative cost of the node itself. We will assume for now that the (◫, ◫) nodes occur at both ends of the path and address the other cases later.

The minimum-cost way to leave a (◫, ◫) node is by connecting to one of the two (□, ▓) nodes. Then we can connect directly back to (◫, ◫), for a total cost of -20. Extending this simple path using a cost 0 length 2 simple cycle, we can create arbitrarily long paths of odd length with cost -20. However, N is even. By using one of the cost $+8$ length 3 simple cycles, we can create paths of any even length ($N \geq 6$) with total cost -12. This produces the paths of class d in the statement of the lemma.

Next, let us consider even-length simple paths that start and end on (◫, ◫) nodes. To achieve an even length, we again need to use at least two ⊡ or ⧇ nodes, and to minimize the cost, we will want to use exactly two. Since we have the beginning and ending (◫, ◫) nodes, exactly two ⊡ or ⧇ nodes, and cannot repeat any (■, ▒) or (□, ▓) node (because we want a *simple* path), we will get a simple path of length 4, 6, or 8.

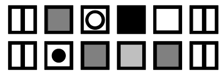
For length 4, we have the simple path version of the simple cycle of length 3 starting and ending with (◫, ◫). As a path, this has total cost -12. However, it contains no (■, ▒) or (□, ▓) node, so cannot be extended to a longer path without increasing the cost.

Next, we can consider paths that include a ⊡ tile. We can indeed achieve a total cost of -12 using the paths



(4.13)

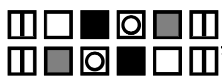
and



(4.14)

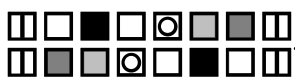
or variations. These paths start with a node, continue to a node, then a node, then a (,) node, and finally a (,) node, before returning to . (Or we could have the same progression from right to left instead.) The difference between the two classes is that in the class represented by (4.14) (class a in the statement of the lemma), one row (the top one in this case) contains both and tiles, whereas in (4.13) (class b in the statement of the lemma), both rows consist of either all and tiles or all and tiles (plus , , and/or). These paths all include (,) and (,) nodes, so can be extended to any even N while maintaining the total cost -12.

There are two more types of simple paths of total cost -12. In these, we use two nodes, and the (,) nodes connect to (,) nodes. Either we have just these six nodes, for instance:



(4.15)

or we also include the two (,) nodes, as in this path:



(4.16)

Both of these types of simple path give class c in the statement of the lemma.

We also need to consider the case where one or both sides are forbidden to have . When one side is forbidden to have , our best strategy is to use the simple path (,) followed by one of the (,) nodes, and then extend using the cost 0 length 2 simple cycle. This path will have a cost of -10. If both sides are forbidden , then we should just use the cost 0 length 2 simple cycle everywhere, for an overall cost of 0.

In the above analysis, we have also learned about the lowest-cost paths of lengths 4 and less. In particular, the lowest cost path of *any* length is the length-1 path (,), which has cost -22. The lowest cost path that does not start and end with (,) nodes has cost -10. The lowest cost path that does not start or end with (,) has cost 0.

At least, this is the case when there are no forbidden pairings in the path. We can now consider what happens when there are forbidden pairings, and show by induction that the lowest cost paths (of whichever structure) do not contain any forbidden pairings. Consider a path of length N with at least one forbidden pairing in an edge. Then the path to left of the forbidden pairing is a shorter path. If it starts and ends with (,) nodes, it could have cost as low as -22. Otherwise, the minimal cost is -10. Similarly for the path on the right. If both left and right paths have (,) nodes adjacent to the forbidden pairing, then there are actually *two* forbidden pairings, and the total cost of the path is at least $+16$. If one path has a (,) node adjacent to the forbidden pairing and the other does not, the minimal cost is at least $(-22)+(-10)+30+6=4$, since $+6$ is the minimal cost to have something horizontally adjacent to ◫. Finally, if neither path has a (◫, ◫) node adjacent to the forbidden pairing, the minimal cost is at least $-20+30=+10$.

If we have a node containing a forbidden vertical pairing, again consider the paths to the left and right of the forbidden node. The cost of the forbidden node is $+60$. If the node is (⊟,⊟), the edges on one side could have total cost as low as -22, although in fact this only happens if the adjacent node is also forbidden. In particular, this cannot happen when the adjacent node is (◫, ◫), so the minimum total cost of the path on one side, plus the cost of the edges connecting to the forbidden node, is at least -32, and that only if the far end of the path is (◫, ◫). Otherwise, it is at least -22. Therefore, the minimal cost of a path which starts and ends with (◫, ◫) and contains a forbidden node is at least -4. The minimal cost of a path which does not have (◫, ◫) on at least one end is at least $+6$. In all cases, we can do better by not including any forbidden pairings. □

Now let us consider the top and bottom rows. Again, we will consider a pair of rows, but with a new formula for costs to take into account the effect of the edge. For the first and second rows, we should consider the following formula:

$$w''(R_1,R_2)=2w(R_1)+w(R_2)+2\sum_{a=1}^{N}w_V(T_{a1},T_{a2})\,. \tag{4.17}$$

We can define $w''(R_{N-1},R_N)$ similarly, counting $w(R_N)$ twice. This formula reflects the appearance of $w(R_1)$ in equation (4.5). Indeed, we have that

$$2w(T)=w''(R_1,R_2)+\sum_{b=2}^{N-2}w'(R_b,R_{b+1})+w''(R_{N-1},R_N)\,. \tag{4.18}$$

Lemma 4.10. *The minimum value of $w''(R_1,R_2)$ is $22-10N$, using a path that has ⊟ everywhere in the top row, and an alternating pattern of either (░, ▓) or (□, ■) in the second row.*

If one corner is required to have a ⊞ tile, the minimum value of $w''(R_1,R_2)$ is $38-10N$. This can be achieved by having ⊟ elsewhere in the top row. In the second row, under the ⊞ tile there is a ◫ tile. Elsewhere the second row alternates between either the pair (░, ▓) or the pair (□, ■).

If the top row begins and ends with ⊞, the minimum value of w'' is $58-10N$, and this can be achieved only if the top row is ⊟ in all other locations. The other row must start and end with ◫, and in between alternate either ■ and □ tiles or ░ and ▓, with a ⊡ tile at one end, just before or after the ◫ tile.

Proof of lemma. Returning to the graph of allowed vertical pairs, the second component now becomes the most favorable: The component with ⊟ at the top can produce a $-\Theta(N)$ total cost for the top pair of rows. Similarly, the component with ⊟ at the bottom can produce a $-\Theta(N)$ total cost for the bottom pair of rows. The component of the graph with ⊟ at the top is pictured in Figure 9. The node and edge weights in Figure 9 are calculated according to equation (4.17).

There are two minimal-cost simple cycles, both of length 2 and cost -20: One contains (⊟, ░) and (⊟, ▓), the other (⊟, □) and (⊟, ■). There are two minimal-cost odd-length cycles, both of length 3 and cost $+18$. They involve (⊞, ◫), (⊟, ⊡), and either (⊟, ▓) or (⊟, □).

The minimal-cost simple paths have length 2 and cost $+2$, e. g., (⊟, ░), (⊟, ▓). This can be extended to even N with a cost -20 length 2 simple cycle to get a total cost of $22-10N$.

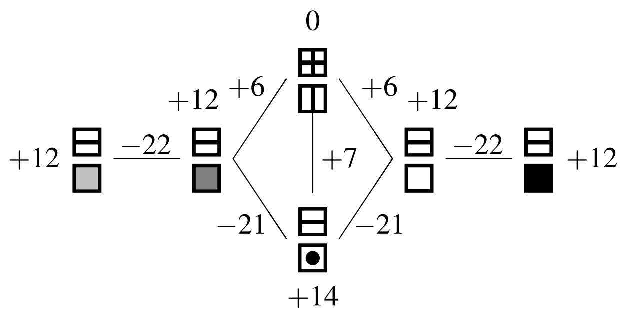


Figure 9: A component of the graph of vertical pairs of tiles. The edge costs are calculated for the w'' formula appropriate for the top two rows.

If we insist that the simple path start at (,), then the minimal-cost length 2 simple path is (,) followed by (,) or (,), which has cost $+18$. The minimal-cost length 3 simple path adds on (,) or (,) on the end, giving cost $+8$. We can than get a minimal-cost path of length N (for N even) by taking the length-2 simple path and extending with a length-2 simple cycle, for total cost $38-10N$.

If we insist that our starting simple path both start and end at (,), our best strategy is to use the length 3 simple cycle as our simple path. As a simple path, it has length 4 and also has cost $+18$. We then get an even length path of total cost $58-10N$. □

As we can see, there is a mismatch between the optimal tilings of the top two rows and the middle pairs of rows. The optimal tilings for a middle row have on both ends, but that would produce forbidden transitions (and a corresponding suboptimal pair of middle rows) if we tried also to put in the optimal tiling for the top two rows.

We wish to apply equation (4.18) to minimize the total cost. We consider various different combinations of tiling the top (and bottom) pairs of rows and middle pairs of rows.

There are three combinations which involve no forbidden pairs.

1. The top and bottom pairs of rows use optimal tilings, and the middle pairs of rows use tilings which do not include on either end. In that case, we find the total cost for Layer 1 is $[2(22-10N)+(N-3)(0)]/2 = 22-10N$.

2. The top and bottom rows each have a in one corner but not in the other. The middle rows have on one end (between the tiles), but not in the other. The optimal tiling subject to these constraints has a cost of $[2(38-10N)+(N-3)(-10)]/2 = 53-15N$.

3. We have tiles in all four corners and the middle pairs of rows use optimal tilings. The total cost is then $[2(58-10N)+(N-3)(-12)]/2 = 76-16N$.

For sufficiently large N, the last choice is optimal.

If we were to use forbidden pairs, we could combine, for instance, an optimal tiling of the top pair of rows with an optimal tiling of most of the middle pairs of rows. However, for that particular combination, we would need to use at least two forbidden pairs, and the cost of doing that is greater than the cost of switching the top pair of rows to a tiling with in both corners. Similarly for other combinations that involve forbidden pairs.

Now let us investigate whether it is actually possible to achieve the cost $76-16N$ using the third combination (in all four corners). By Lemma 4.9, we have vertical lines of tiles on the left and right sides of the grid, and with a path of or tiles somewhere in between them. The path is diagonal everywhere, but could potentially move back and forth to the left and right. If the dividing line is adjacent at some point to an end tile, it is a tile; otherwise, it is a tile. The dividing line splits the remaining locations into two or more connected components, each of which consists of rows of alternating and tiles and rows of alternating and tiles, with the two types of rows alternating as well.

In summary, we have at all four corners, everywhere else in the top and bottom rows, and everywhere on the left and right sides, except at the corners. There is exactly one or tile in each row. In the second and $(N-1)$th rows, it must be a tile and must be located in either the second or $(N-1)$th column. Elsewhere, the or tiles from two adjacent rows must be located diagonally from each other. Since N is even, it is not possible to do all of this if the tiles from the top and bottom rows are in the same column. Therefore, the and tiles must form a diagonal line reaching from one corner (e. g., the $(2,2)$ location) to the opposite corner (e. g., the $(N-1,N-1)$ location) of the interior. The ends of the line are tiles, and the rest of the diagonal line is composed of tiles. On each side of the line, the interior of the grid is tiled by rows alternating between and tiles and and tiles. Columns 2 and $N-1$ contain only and tiles, as well as and .

We thus get 8 possible minimal cost solutions for Layer 1. There is the arrangement of Figure 7, and three rotations of it. These correspond to cases b and d in Lemma 4.9. Alternatively, we can take the upper left corner or lower right corner of Figure 7 and rotate it 180° around the center of the grid to fill in the other corner. There is one rotation for each of those solutions as well, with the diagonal line going from the upper left to the lower right instead. These correspond to cases a and c in Lemma 4.9. The first four solutions, rotations of Figure 7, have a local breaking of reflection symmetry in the vicinity of the diagonal line: In the orientation of Figure 7, immediately to the left of and above the diagonal line, we have and tiles, whereas immediately to the right of and below the diagonal line, we have and tiles. In contrast, while the other four solutions also break vertical and horizontal reflection symmetry, there is no local, translationally-invariant rule anywhere in the grid that can allow us to distinguish the directions.

Layer 2 Layer 2 will have only 3 types of tile: , , . The tiling rules will only specify permitted or forbidden pairs of adjacent tiles; forbidden pairs have a cost of $+30$, as before, and permitted pairs have a cost of 0. When the Layer 1 tile is or , the Layer 2 tile must be . For any other Layer 1 tile, any of the three Layer 2 tiles is possible.

For the vertical tiling rules, we only allow each type of tile to be adjacent to itself. Horizontally, each type of tile is forbidden to be adjacent to itself. Thus, to avoid any forbidden pairs of tiles, Layer 2 must consist of vertical lines, all of one kind of tile, and no two adjacent lines can be the same. Our goal is to have the three types of tile cycle: followed by followed by , and then back to . Then by looking at Layer 2 for any horizontally adjacent pair of locations in the grid, we can distinguish left from right.

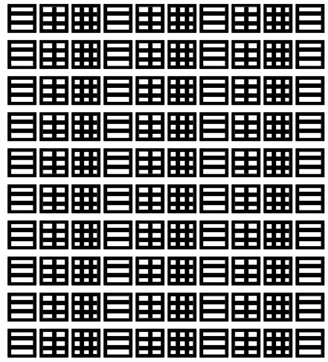
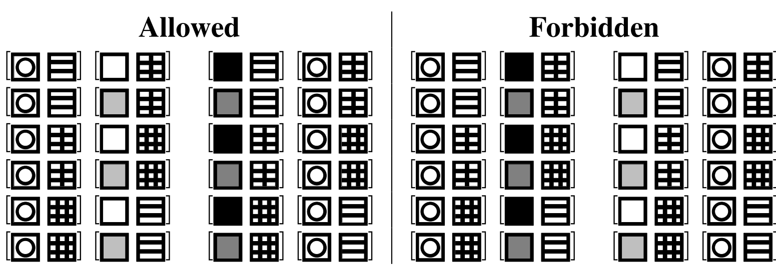


Figure 10: The permitted tiling of Layer 2 for a 10×10 grid when Layer 1 is the optimal tiling of Figure 7. Notice that at every location, it is possible to distinguish left and right by looking at the adjacent tiles.

To achieve this, we determine whether two different Layer 2 tiles can be adjacent horizontally depending on the Layer 1 tiles underlying them. If neither of the Layer 1 tiles is a or , any pair of Layer 2 tiles is permitted. If one of the Layer 1 tiles is a , , or tile (although the latter two are forbidden in Layer 1 to be horizontally adjacent to or), then any pair of Layer 2 tiles is permitted. It is forbidden in Layer 1 to have two or tiles adjacent, but if it happens, we allow any pair of Layer 2 tiles. Otherwise, the following combinations are allowed. We use brackets to indicate the pair of Layer 1 and Layer 2 tile, so [ ] indicates a on Layer 1 and on Layer 2 in a given location. In all of the following pairs, we use for one of the Layer 1 tiles, but we have precisely the same rules if is substituted for .



(4.19)

If Layer 1 is the tiling of Figure 7, then the middle $N-2$ columns of Layer 2 will, according the rules of (4.19), cycle between columns of tiles, tiles, and tiles, in that order from left to right. Columns 1 and N, since Layer 1 contains tiles there, must be tiles on Layer 2. Since adjacent columns must have different Layer 2 tiles, column 2 cannot be tiles. If column 2 contained tiles, since $N \equiv 1 \bmod 3$, then column $N-1$ would contain tiles, which is forbidden, since column N also contains tiles. Thus, column 2 contains tiles and column $N-1$ contains tiles, so the whole grid cycles between the three types of tiles for Layer 2, as pictured in Figure 10. Similarly if Layer 1 is a rotation of Figure 7.

On the other hand, if Layer 1 is arranged according to one of the other optimal tilings, there will be no allowed tiling of Layer 2. This is because in two adjacent rows, we will have a configuration such as

$$\begin{array}{cc} ■ & ◙ \\ ◙ & ■ \end{array} \tag{4.20}$$

There is no way to tile Layer 2 for these four locations consistent with both the rules of (4.19) and the constraint that only identical tiles can be vertically adjacent in Layer 2. Thus, in order to achieve a cost of $76-16N$, Layer 1 must be in a tiling corresponding to one of the four rotations of Figure 7.

Layer 3 Layer 3 is very similar to Layer 2. There are again 3 kinds of tiles: ▥, ╫, and ⌗. When the Layer 1 tile is ⊟, the Layer 3 tile must be ▥; otherwise, any Layer 3 tile is allowed. The adjacency rules are effectively the same as Layer 2, but with horizontal and vertical exchanged, and with ▥, ╫, and ⌗ substituted for ▤, ▦, and ▩, respectively. Thus, the only permitted tiling of Layer 3 has rows consisting of a single type of tile, with the three kinds of rows cycling.

Main layers To tile the main layers, we just use the rules from Section 3. Those rules distinguish between up and down and between left and right, so to achieve that, we look at the tiles in Layers 2 and 3. If Layer 2 is a ▤ for one location and ▦ for the second location, we consider the first location to be to the left of the second location. Similarly, we consider a ▦ Layer 2 tile to be to the left of a ▩ Layer 2 tile, and a ▩ Layer 2 tile to be to the left of a ▤ Layer 2 tile. We consider a ▥ Layer 3 tile to be above a ╫ Layer 3 tile, a ╫ Layer 3 tile to be above a ⌗ Layer 3 tile, and a ⌗ Layer 3 tile to be above a ▥ Layer 3 tile. We can set the corner boundary conditions for the main layers by looking at Layer 1: When the Layer 1 tile is a ⊞ tile, the main layer tile must be a corner tile.

If there is a valid tiling of the main layer, which occurs when the instance is a "yes" instance, then we achieve an overall cost $76-16N$. If the instance is a "no" instance, then if Layer 1 has an optimal arrangement, and Layers 2 and 3 are in the allowed tiling consistent with Layer 1, then there cannot be a valid tiling of the main layer. Thus, for a "no" instance, the cost must be greater than $76-16N$. □

4.5.2 Periodic boundary conditions

Now we turn to WEIGHTED TILING with reflection symmetry and periodic boundary conditions. In this case, we are able to prove results for both constant and linear cost functions:

Theorem 4.11. *For WEIGHTED TILING with reflection symmetry and periodic boundary conditions:*

- *When the cost function $p(N)$ is a constant c, independent of N, the problem is in* P. *In particular, 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.*

- *When the cost function $p(N)$ is linear in N, the problem is* NEXP*-complete.*

- *When N is even, the problem is in* P *for any cost function $p(N)$.*

Proof. **Constant cost function $p(N) = c$, some constant independent of N** 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 $w(R_a) = \sum_b w_H(T_{ab}, T_{a(b+1)})$, where T_{ab} is the tile in location (a,b), and let $w(R_a, R_{a+1}) = \sum_b w_V(T_{ab}, T_{(a+1)b})$. Then $c' = \sum_a (w(R_a) + w(R_a, R_{a+1}))$. Notice that if we duplicate two adjacent rows, say R_a and R_{a+1}, then the total cost becomes $c' + w(R_a) + w(R_{a+1}) + 2w(R_a, R_{a+1})$ (because of the reflection symmetry). Now,

$$\sum_{a=1}^{N} [w(R_a) + w(R_{a+1}) + 2w(R_a, R_{a+1})] = 2 \sum_{a=1}^{N} [w(R_a) + w(R_a, R_{a+1})] = 2c' . \tag{4.21}$$

Thus, there must exist some a for which

$$w(R_a) + w(R_{a+1}) + 2w(R_a, R_{a+1}) \leq 2c'/N . \tag{4.22}$$

When c' is a constant, for $N > 2c'$, that means there is some a for which

$$w(R_a) + w(R_{a+1}) + 2w(R_a, R_{a+1}) \leq 0 , \tag{4.23}$$

since the weights are integers.[3] Then we can duplicate rows R_a and R_{a+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 4.6 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 $p(N)$ be a constant. It fails if $p(N)$ grows at least linearly with N.

Even N When N is even, we can easily compute the exact minimal cost achievable. Consider all possible 2×2 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 2×2 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 2×2 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.

Linear cost function The construction uses similar ideas to the case with open boundary conditions, but differs in some details.

[3] If we generalize the definition of WEIGHTED TILING to allow rational weights, we can simply rescale to get integer weights. If we allow irrational weights, this argument fails, but it is possible to prove the same result using the ideas presented in the even N case below.

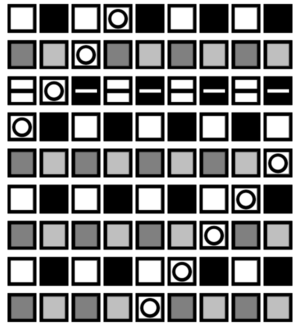


Figure 11: An optimal tiling for the periodic boundary condition case with $N = 9$.

The argument now uses 4 layers of tiles, plus a main layer implementing the tiling rules of Section 3. The first three layers work in much the same way as the three layers discussed in the previous proof for WEIGHTED TILING with open boundary conditions. The fourth layer defines borders that can be used to set the boundary conditions in the main layer. We will consider odd N, with N divisible by 3 (so $N \equiv 3 \bmod 6$). The allowed cost is $6N - 2$.

Layer 1 Layer 1 uses tiles , , , , , , and . The tiles , , , , and serve much the same purpose as in the proof for open boundary conditions, and indeed have the same adjacency rules between them. Only the adjacency rules for and are new.

The desired optimal tiling for Layer 1 has, for all but one row, a single tile interrupting an alternating line of either and or and (with the two types of row alternating). The remaining row consists of alternating and tiles, also interrupted by one tile. The tiles form a diagonal line circling the torus. An example of the optimal tiling is given in Figure 11.

To achieve this, we use the adjacency rules given in Table 11.

The only 2×2 squares with total cost 0 involve one each of , , , and . Since they must alternate colors both vertically and horizontally, but N is odd, it is not possible to fill the whole torus with such squares. Indeed, if we wish to avoid forbidden pairings, any region containing only , , , and tiles cannot include any path which is topologically non-trivial on the torus. Such a path would circle the torus either vertically, horizontally, or both, and would therefore produce inconsistent constraints on the tile types. Any tiling must therefore contain enough adjacent pairs of tiles in the grid with nonzero cost to block the non-trivial cycles. Forbidden pairings have a high cost, so the cheapest way to block the non-trivial cycles is with , , and tiles. To block all kinds of non-trivial cycles, we need at least $2N - 1$, , or tiles in the tiling.

Each included in the tiling has a net cost of 4, since for any tile (other than another), a placed adjacent to it in any direction costs exactly 1 more than any other allowed tile in the same position. We can account each or tile at an overall cost of 2, since that is the minimum cost to place tiles both above and below it. Therefore, and tiles are cheaper than tiles. However, and tiles are only allowed to be horizontally adjacent to each other or a tile. Therefore, the and tiles must form horizontal lines. One such line can act as one of the two topologically non-trivial cycles, but a second such line is not helpful. Furthermore, and must alternate horizontally, which means we cannot go

Horizontal tiling rules

	30	0	30	30	30	30	1
	0	30	30	30	30	30	1
	30	30	30	0	30	30	1
	30	30	0	30	30	30	1
	30	30	30	30	30	0	1
	30	30	30	30	0	30	1
	1	1	1	1	1	1	30

Vertical tiling rules

	30	30	1	30	1	30	2
	30	30	30	1	30	1	2
	1	30	30	30	30	0	1
	30	1	30	30	0	30	1
	1	30	30	0	30	30	1
	30	1	0	30	30	30	1
	2	2	1	1	1	1	30

Table 11: The tiling weights for layer 1 for WEIGHTED TILING with reflection symmetry and periodic boundary conditions. Since there is reflection symmetry, the horizontal and vertical tiling weight matrices are symmetric.

		Layer 4 tile											
		N	N	N	N	N	N	N	N	N	Y	N	N
		N	N	N	N	N	N	N	N	N	Y	N	N
Layer 1		Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
tile		Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
		Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
		Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
		Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y

Table 12: The compatibility rules between layer 1 tiles and layer 4 tiles for WEIGHTED TILING with reflection symmetry and periodic boundary conditions.

all the way around a horizontal cycle with just them. Therefore, the horizontal line must contain at least one ; the cost will be minimal if there is exactly one.

Thus, to minimize the cost, we must have a configuration which contains a large region of , , , and tiles, arranged to avoid forbidden pairings, plus a horizontal line containing $N-1$ alternating and tiles, and N additional tiles, for a total cost of $2(N-1)+4N=6N-2$. Furthermore, there must be exactly one tile in each row (since there are only N of them, and each row must have at least one in order to get the parity right). The tiles cannot be adjacent vertically, and they must form a continuous path (in order to form a second topologically non-trivial cycle), so they are adjacent diagonally. Because N is odd, the only way that the tiles can form a closed path is therefore for them to form a single diagonal line. In short, to achieve a cost $6N-2$, we must have a configuration much like that of Figure 11.

Layers 2 **and** 3 Layers 2 and 3 work almost the same way as for the open boundary condition case. For the purpose of the Layer 2 and Layer 3 rules, a Layer 1 tile is treated the same way as a Layer 1 tile, and a Layer 1 tile is treated the same way as a Layer 1 tile. As before, Layer 2 will cycle between columns of , , and , and Layer 3 will cycle between rows of , , and tiles. Since N is divisible by 3, it is possible to do this consistently, and there are no other configurations consistent with an optimal Layer 1 tiling. However, unlike the open boundary conditions case, there is nothing to fix which columns have and which rows have , so there are three possible Layer 2 tilings and three possible Layer 3 tilings which are consistent with any optimal Layer 1 tiling.

Layer 4 Layer 4 is used to set the boundaries of the square $(N-1)\times(N-1)$ region used in the main layer to implement the construction of Section 3. It uses the tiles , , , , , , , , , , , and . We consult Layer 1 to determine the locations of boundaries, and Layers 2 and 3 to determine the orientation left/right and up/down. The tiling weights are given in Table 13. We also have the rules in Table 12 which specify which pairs of layer 4 tiles and layer 1 tiles can be in the same location.

Suppose we have an optimal tiling of Layer 1. Then there is a row which contains and , with one . The locations with or on Layer 1 must have on Layer 4, but the location cannot, so the only remaining possibility for that location is a . Since must have adjacent to it horizontally, and

	Tile on right											
	0	30	30	30	30	0	30	30	30	30	30	30
	30	0	30	30	30	30	30	0	30	30	30	30
	30	30	30	30	30	30	30	30	30	30	0	30
	30	30	30	30	30	30	30	30	0	30	30	30
Tile	0	30	30	30	30	0	30	30	30	30	30	30
on	30	30	30	30	30	30	30	30	30	30	0	30
left	30	0	30	30	30	30	30	0	30	30	30	30
	30	30	30	30	30	30	30	30	30	30	0	30
	30	30	0	30	30	30	30	30	0	30	30	30
	30	30	30	30	30	30	30	30	30	0	30	0
	30	30	30	0	0	30	0	30	30	30	30	30
	30	30	30	30	30	30	30	30	30	0	30	30

	Tile on top											
	30	30	30	30	30	30	30	30	30	0	30	30
	30	30	30	30	30	30	30	30	0	30	30	30
	30	30	0	30	30	0	30	30	30	30	30	30
	30	30	30	0	0	30	30	30	30	30	30	30
Tile	30	30	30	30	30	30	30	30	30	0	30	30
on	30	30	30	30	30	30	30	30	30	0	30	30
bottom	30	30	30	0	0	30	30	30	30	30	30	30
	30	30	0	30	30	0	30	30	30	30	30	30
	0	30	30	30	30	30	30	30	0	30	30	30
	30	0	30	30	30	30	0	0	30	30	30	30
	30	30	30	30	30	30	30	30	30	30	0	0
	30	30	30	30	30	30	30	30	30	30	0	30

Table 13: The tiling weights for layer 4 for WEIGHTED TILING with reflection symmetry and periodic boundary conditions. Even though the underlying rules have reflection symmetry, we have presented them in a way without reflection symmetry. The distinction between "left" and "right" in a horizontally adjacent pair is provided by the corresponding pair of layer 2 tiles, and the distinction between "top" and "bottom" for a vertically adjacent pair is provided by layer 3.

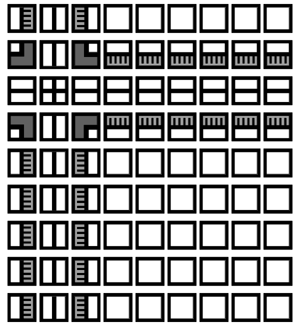


Figure 12: The arrangement of layer 4 when the Layer 1 tiling is given by Figure 11.

there can be no other tiles on Layer 4, this is the only location on Layer 4 that contains a tile. The rest of the column containing the must thus be all tiles. That is, there is a vertical line of tiles intersecting a horizontal line of tiles at a location. There can be no other , , or elsewhere in Layer 4.

We consider the remaining tiles to be the "interior" of the tiling, defining an $(N-1)\times(N-1)$ grid. The tiles , , and will mark the upper boundary of the interior, , , and mark the right edge of the interior, , , and mark the bottom row, and , , and indicate the leftmost column of the interior. The corners of the interior are indicated by , , , and . The remaining interior tiles are . The resulting tiling pattern is given in Figure 12.

Main layers In the main layers, we again define directions, breaking the reflection symmetry by looking at Layers 2 and 3. We implement the protocol of Section 3 on an $(N-1)\times(N-1)$ grid defined by the interior locations determined by Layer 4. If the Layer 4 tile is a , , or , the main layer tile must be an extra tile not used in the main set. The extra tile type can only be in a location that has one of those three tile types on Layer 4, and it can be adjacent to any other main layer tile. Locations on Layer 4 that have a , , , or tile must have a corner tile in the main layer. The other Layer 4 tiles can correspond to any main layer tile. Thus, when Layer 1 has an optimal tiling, and Layers 2 – 4 have no forbidden pairings, the main layer only has a tiling without forbidden pairings if the universal Turing machine M accepts on input $f_{BC}(N-1)$. That is, there is an overall tiling of cost $6N-2$ iff the problem is a "yes" instance; otherwise, the cost is higher. $\square$

4.6 Weighted tiling with rotation symmetry

If we have *rotation symmetry*, we have reflection symmetry and also $w_V = w_H$. We therefore drop the subscript in this case and use a single function $w : T \times T \to \mathbb{Z}$. In this case, even the WEIGHTED TILING problem is easy:

Theorem 4.12. *For WEIGHTED TILING with rotation symmetry, we have the following results:*

- *With 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 $p(N)$.)*

- *With 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 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 $N \geq N_o$.*

Proof. **Rotation symmetry and open boundary conditions** This case is essentially trivial. We find the minimal weight w 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. The resulting cost is $2N(N-1)w$.

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 6). 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 since we are assuming $p(N) = o(N^2)$. If there exists a pair of tiles with negative weight, 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-weight cycle of odd length containing a zero-cost pairing. (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 every pair of tiles is positive, then it is certainly not possible to tile an $N \times N$ grid when $2N(N-1) > p(N)$. 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 $2N(N-1) - 8 \geq 8w - p(N)$.

The only potentially 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. We can assume $p(N) = c$, a constant, as we can always achieve a constant cost

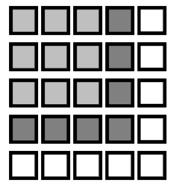


Figure 13: The shaded tiles are an upper-left 4-square. The tiles in the darker shade are the upper-left 4-border.

$8w$ using the strategy above (0 cost pairs everywhere but the corners). We will show that if there exists a tiling of constant cost, the vicinity of the four corners must satisfy certain conditions. Then by running through the possible configurations near the corners, we can find a minimum-weight tiling.

Consider the graph whose nodes are tile types t_i and edges correspond to pairings of tiles with 0 cost. Let Z_α 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_α, with only a constant number of different tiles, and that indeed it is sufficient to place all the non-Z_α 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 define the *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 *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 13. We say that a square is *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_α.

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_α. 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:

Definition 4.13. Suppose we have two k-squares I_1 and I_2 that are located at different corners of the grid, with borders from Z_α. We say they are *compatible for N and Z_α* if, given tile t_i on the border of I_1 and

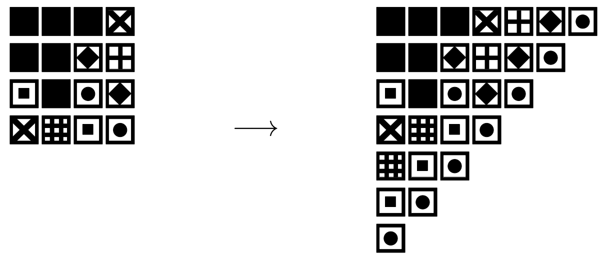


Figure 14: Extending a square to make a corner triangle. The black tiles have nonzero cost to be adjacent to any other tile. All other pairings that occur in the figure are zero cost.

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_α* 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 α, valid squares in the four corners I_1, I_2, I_3, and I_4 which are compatible for N and Z_α 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 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 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_α. The converse is also true:

Lemma 4.14. *Let I_1, I_2, I_3, and I_4 be four squares which are compatible for N and Z_α, 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_α 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 14. Finally, we can add additional diagonal layers to make sure the outer diagonal border is any particular tile from Z_α, and to increase the size of the triangle as much as desired.

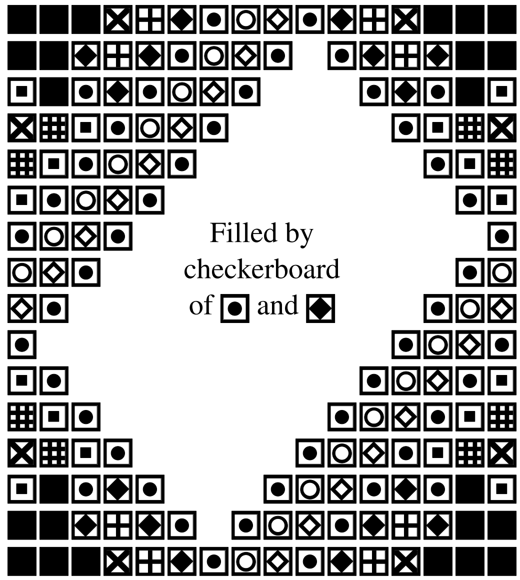


Figure 15: A 16×16 grid tiled by extending triangles on the four corners.

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_α, then their corresponding triangles remain compatible when extended as described above, at least when N is large. 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 6 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_α which has a zero-cost pairing with t_i. See Figure 15 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. □

5 The quantum one-dimensional chain

5.1 Preliminaries

In the 2-dimensional classical tiling reduction, we used a set of uniform rules to encode a dynamic computation in a static object, the placement of tiles on a grid. Each configuration of the Turing Machine computation was encoded in a row of the grid of tiles. Similarly, in the quantum case, we encode a quantum computation in the ground state of a translationally-invariant Hamiltonian. However, in the quantum case, the different steps of the computation are represented in quantum superposition. Previous QMA reductions ([22, 1]) have used the circuit model to represent a generic polynomial-time quantum computation, but we use instead a quantum Turing Machine because it is more amenable to translationally-invariant systems. Encoding a quantum circuit requires encoding individual gate types at different particles in the system which is not a problem with position-dependent terms.

Quantum Turing machines were first introduced in [8] and further developed in [5] and are a natural extension of the classical ones. In a classical Turing Machine, a pair (q,a) transitions to a unique move, specified by triplet $(L/R,q',b)$. In a quantum Turing Machine, a pair (q,a) transitions to a superposition of moves. Thus, at any point in time, the state of the Turing Machine is a quantum superposition of classical configurations.

The computational process encoded in the quantum case is very similar to the classical tiling reduction: we make use of a binary counting Turing machine M_{BC} to translate the length of the chain into a binary string x and then use x as the input to a quantum Turing Machine that computes a $\mathsf{QMA}_{\mathsf{EXP}}$-complete language. Because we are working with quantum systems, we will require that M_{BC} be reversible, meaning each configuration of the Turing machine has a unique predecessor in the computation. Bernstein and Vazirani [5] have shown that any deterministic Turing machine can be made reversible with some additional but not significant overhead. We can still assume that there is a function $f_{BC} : \mathbb{Z} \to \{0,1\}^*$ such that for some constant N_0 and every $N \geq N_0$, if M_{BC} runs for N steps, then the string $f_{BC}(N)$ will be written on the tape with the rest of the tape blank. Moreover there are constants c_1 and c_2 such that if n is the length of the string $f_{BC}(N)$ and $N \geq N_0$, then $2^{c_1 n} \leq N \leq 2^{c_2 n}$. We will also assume that for any binary string x, we can compute N such that $f_{BC}(N) = x$ in time that is polynomial in the length of x.

We can reduce any language in $\mathsf{QMA}_{\mathsf{EXP}}$ to a language L that is accepted by a verifier who uses a witness of size $2^{c_1 n}$ and whose computation lasts for $2^{c_1 n}$ steps, where n is the length of the input. This is the same reduction used in the classical case, in which the input is padded to length $|x|^k/c_1$. We can then use standard boosting techniques to assume that the probability of acceptance is at least $1-\varepsilon$ for a "yes" instance or at most ε for a "no" instance, with $\varepsilon = 1/\mathrm{poly}(N)$ [22]. Thus, we can assume that our verifier is 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 $1-\varepsilon$, 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 $|\psi\rangle$, V accepts with probability at most ε, then the ground state energy of the system is at least $p(N)+1/q(N)$. According to [5], we can assume without loss of generality that the Turing machine V has a one-way infinite tape and that the head starts in designated start state q_0 at the left-most end of the tape. We will also assume that on input x, after $2^{c_1|x|}$ steps, the Turing machine is in an accepting or rejecting state and the head is again at the left-most end of the tape. We will also assume that the witness will be stored in a parallel track with the left-most qubit in the left-most position of the tape.

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

Definition 5.1. A standard basis state is *bracketed* if the left-most particle is in state $<$, the right-most particle is in state $>$, and no other particle in the chain is in state $<$ or $>$. $\mathcal{S}_{br}$ is the space spanned by all bracketed states.

The Hamiltonian we describe below will be closed on the space $\mathcal{S}_{br}$. We will restrict our attention for now to $\mathcal{S}_{br}$ and add a term later in Section 5.9 that gives an energy penalty to any state outside $\mathcal{S}_{br}$. The rest of the particle 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 a sort of 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 column represents the state of an individual particle.

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

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:

$$\frac{1}{2}\left(|ab\rangle\langle ab| + |cd\rangle\langle cd| - |ab\rangle\langle cd| - |cd\rangle\langle ab|\right).$$

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 $ab \to cd$. Type II terms are also referred to as *transition rules*. We will say that configuration A transitions into configuration B by rule $ab \to cd$ 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.

5.2 Outline of the construction

Section 5.3 describes the rules for Tracks 1 and 2 which maintain the clock. There is a special set of particle states for Track 1 called *pointer* states. The rules enforce that there is exactly one particle in a pointer state on Track 1 at each point in time. The pointer moves back and forth from one end of the chain to the other via the transition rules. The clock progresses in three phases and the pointer states on Track 1 will be different for the different phases. The contents of Track 2 control the number of iterations of the pointer particle on Track 1 as well as the transitions from one phase to another. This part of the proof is almost entirely classical as it consists of establishing a set of combinatorial properties about classical states and the transitions between them. The only quantum part is at the end in Lemma 5.6 in which these combinatorial properties are related to the spectral gap of the Hamiltonian.

In each of the three phases of the clock, the pointer particle triggers different actions on the other tracks. The first phase (the Initialization Phase) is used to check that the Turing Machine tracks are properly initialized. This is described in Section 5.4. In the second phase (the Counting Phase), each iteration of the pointer state on Track 1 triggers a step of the Turing Machine M_{BC} and in the third phase it triggers the execution of a step of the quantum verifier Turing Machine V. These are described in Sections 5.5 and 5.6.

Section 5.7 puts these different components together and describes the energy of the ground state from all the terms described so far. 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. The pointer symbol on Track 1 interacts with the other four tracks to trigger steps of the Turing Machines. 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. We need to have illegal pairs that cause clock states other than those described above to have an energy cost. As is the case in other such proofs (see for example [1]), 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. We therefore establish that any low energy state must correspond to a legal computation beginning from the correct start state.

In Section 5.8 an additional penalty is added for any state reflecting a non-accepting computation. In Section 5.9, we add a final term which penalizes any state outside $\mathcal{S}_{br}$. This concludes the proof of Theorem 2.5.

5.3 The clock tracks

Most of this section rests on combinatorial (non-quantum) arguments in the following sense. Consider a graph such that each node corresponds to a standard basis state for Tracks 1 and 2. For each pair of states A and B, add a directed edge from A to B if there is a single transition rule which transforms A to B in the forward direction. We will effectively show that each node corresponding to a legal state is in a connected component that is a path. Moreover, there is exactly one such path that has no illegal states. In every other such path, the minimum distance to an illegal node is at most $2N$. Lemma 5.6 then uses standard arguments to connect these properties to the structure of the ground state and the spectral gap.

Allowed clock states 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. A *regular expression* is an expression that specifies a set of strings by denoting a sequence of operations that can be performed to construct a string in the set. These operations are restricted to concatenation, the OR function or repetition.

Lemma 5.2. *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. □

There will be four types of states in Track 1: $\textcircled{\triangleright}$, $\textcircled{\triangleleft}$, $\circledcirc$, and $\bigcirc$. (However, the $\textcircled{\triangleright}$ and $\textcircled{\triangleleft}$ states each come in multiple varieties—see below for the details.) Illegal pairs are used to enforce that the state of Track 1 is always a substring of a string of the form $\textcircled{<}\circledcirc^*(\textcircled{\triangleright}\|\textcircled{\triangleleft})\bigcirc^*\textcircled{>}$. (The $\|$ denotes the OR operator.) Given our restriction (which will be justified in Section 5.9) to $\mathcal{S}_{br}$, the space spanned by all bracketed states, this implies that the state is always a string of the form $\textcircled{<}\circledcirc^*(\textcircled{\triangleright}\|\textcircled{\triangleleft})\bigcirc^*\textcircled{>}$. The set of all such strings forms a *local language*, since it defined by local constraints.

There is one *pointer* symbol on Track 1 (of type $\textcircled{\triangleright}$ or $\textcircled{\triangleleft}$) that shuttles back and forth between the left end and the right end and operates as a second hand for our clock. The triangle points in the direction that the symbol is moving. These states are also sometimes called *control* states since they trigger actions on the other tracks. We call one round trip of the pointer symbol 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: $\textcircled{0}$, $\textcircled{1}$, $\textcircled{2}$, and $\textcircled{\bar{0}}$ and $\textcircled{\bar{1}}$. We will have illegal pairs that will enforce that any legal configuration on Track 2 must be a substring of $\textcircled{<}\textcircled{1}^*(\textcircled{\bar{0}}\textcircled{0}^*\|\textcircled{\bar{1}}\textcircled{2}^*)\textcircled{>}$. Additionally, we will add terms which will force the ground state to have a $\textcircled{<}$ symbol on the left end and a $\textcircled{>}$ symbol on the right end. Thus, Track 2 will actually be of the form $\textcircled{<}\textcircled{1}^*(\textcircled{\bar{0}}\textcircled{0}^*\|\textcircled{\bar{1}}\textcircled{2}^*)\textcircled{>}$.

The Track 2 states are partitioned into two phases. The first phase is called the *Counting Phase* and consists of all $N-2$ of the states of the form $\textcircled{<}\textcircled{1}^*\textcircled{\bar{0}}\textcircled{0}^*\textcircled{>}$. The second phase is the *Computation Phase* and consists of all $N-2$ of the states of the form $\textcircled{<}\textcircled{1}^*\textcircled{\bar{1}}\textcircled{2}^*\textcircled{>}$ states. The $\textcircled{<}\textcircled{1}^*\textcircled{\bar{0}}\textcircled{0}^*\textcircled{>}$ states are ordered according to the number of particles in state $\textcircled{1}$ and the $\textcircled{<}\textcircled{1}^*\textcircled{\bar{1}}\textcircled{2}^*\textcircled{>}$ states are ordered

Phase	Function	Track 1 state	Track 2 state	Number of steps
Initialization	Check start state	$\textcircled{<}\circledcirc^*(\textcircled{0\triangleright}\,\|\|\,\textcircled{\triangleleft 0})\bigcirc^*\textcircled{>}$	$\textcircled{<}\textcircled{\bar{0}}\textcircled{0}^*\textcircled{>}$	$2(N-2)$
Counting	Run M_{BC}	$\textcircled{<}\circledcirc^*(\textcircled{1\triangleright}\,\|\|\,\textcircled{\triangleleft 1})\bigcirc^*\textcircled{>}$	$\textcircled{<}\textcircled{1}\textcircled{1}^*\textcircled{\bar{0}}\textcircled{0}^*\textcircled{>}$	$(N-2)(2N-5)$
Computation	Run V	$\textcircled{<}\circledcirc^*(\textcircled{2\triangleright}\,\|\|\,\textcircled{\triangleleft 2})\bigcirc^*\textcircled{>}$	$\textcircled{<}\textcircled{1}^*\textcircled{\bar{1}}\textcircled{2}^*\textcircled{>}$	$(N-2)(2N-5)$

Table 14: The different clock phases in the quantum construction on a line.

according to the number of particles in state $\textcircled{2}$. The state immediately after $\textcircled{<}\textcircled{1}^*\textcircled{\bar{0}}\textcircled{>}$ in the ordering is $\textcircled{<}\textcircled{1}^*\textcircled{\bar{1}}\textcircled{>}$. (See Figure 17 for an example).

Track 1 transitions Each of the pointer states for Track 1 will come in three varieties: $\textcircled{0\triangleright}$ and $\textcircled{\triangleleft 0}$ will be used during the initial minute of the clock when it is in state $\textcircled{<}\textcircled{\bar{0}}\textcircled{0}^*\textcircled{>}$ and will be used to check initial conditions on the other tracks. $\textcircled{1\triangleright}$ and $\textcircled{\triangleleft 1}$ will be used during the counting phase and $\textcircled{2\triangleright}$ and $\textcircled{\triangleleft 2}$ will be used during the computation phase. $\textcircled{1\triangleright}$ and $\textcircled{2\triangleright}$ will be used to trigger different actions on the other tapes. The symbol $\textcircled{\triangleright}$ will be used to generically denote one of the three symbols, $\textcircled{0\triangleright}$, $\textcircled{1\triangleright}$ or $\textcircled{2\triangleright}$. Similarly, the symbol $\textcircled{\triangleleft}$ will be used to generically denote one of the three symbols, $\textcircled{\triangleleft 0}$, $\textcircled{\triangleleft 1}$ or $\textcircled{\triangleleft 2}$. Every time the $\textcircled{1\triangleright}$ 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 $\textcircled{2\triangleright}$ 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.

The transition rules move the pointer states in the direction in which they are pointing:

1. $\textcircled{\triangleright}\bigcirc \rightarrow \circledcirc\textcircled{\triangleright}$, $\circledcirc\textcircled{\triangleleft} \rightarrow \textcircled{\triangleleft}\bigcirc$: in the forward direction, pointers move in the direction they are pointing. In the reverse direction, they move in the opposite direction.

2. $\textcircled{\triangleright}\textcircled{>} \rightarrow \textcircled{\triangleleft}\textcircled{>}$, $\textcircled{<}\textcircled{\triangleleft} \rightarrow \textcircled{<}\textcircled{\triangleright}$: pointers change direction when they hit an endpoint.

Note that the underlying Hamiltonian terms that produce the transitions are symmetric, so any transition can occur in either direction. We choose one direction to be the "forward" direction and one to be the "reverse" direction, as described above, in order to clarify the structure of protocol.

A single iteration for the pointer symbol on Track 1 is depicted in Figure 16. Note that the $\textcircled{\triangleright}$ 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 $\textcircled{\triangleleft}$ 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 legal configuration for Track 1 has exactly one transition rule that applies in the forward direction and one that applies in the reverse direction. The pointer 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. Every iteration has $2(N-2)$ distinct states and $2(N-2)$ transitions.

Track 2 transitions Each iteration of Track 1 causes one change in the configuration on Track 2 which acts then as a minute hand for the clock. The first phase of transitions for Track 2 are depicted in Figure 17. We need to describe how the different Track 1 pointer symbols trigger transitions on Track

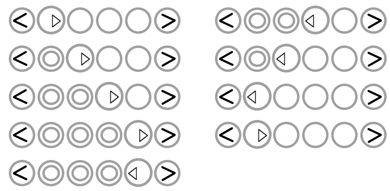
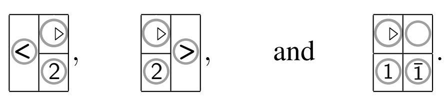


Figure 16: One iteration for Track 1. The figure should be read in column form from top to bottom, first the left column, then the right.

Figure 17: The first phase for Track 2 for a six-particle system. The figure should be read in column form from top to bottom.

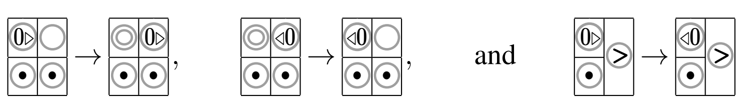
2. The transitions for Track 1 will remain as described in that the transition ▷◯ → ◎▷ will always cause a transition from one of the ▷ symbols to one of the ▷ symbols, but we need to specify which ▷ symbols are used in order to ensure that forward and backwards transitions remain unique for each standard basis state.

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

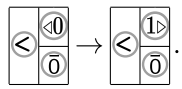


We will use the symbol ⊙ to denote a variable state. Thus [▷, ⊙] is used to denote any state in which the Track 1 state is ▷.

Clock states for the initialization The initial iteration of the second hand is demonstrated for Tracks 1 and 2 on a chain of length six in Figure 18. When Track 1 has a 0▷ or ◁0 symbol, we want to ensure that Track 2 is in state ⧀ $\bar{0}$ ⓪* ⧁. Therefore, we disallow [0▷, ⊙] and [◁0, ⊙] for any ⊙ that is not ⓪ or $\bar{0}$. We have the usual transitions that advance the Track 1 pointer:



These happen regardless of the values on the other tracks and do not change the values on the other Tracks. The state $\textcircled{<}[\textcircled{0\rhd}, \textcircled{\bar{0}}]$ does not have a transition in the reverse direction since this only occurs in the initial state. Finally, we have the transition



The presence of the $\textcircled{\bar{0}}$ on Track 2 specifies that $\textcircled{1\rhd}$ should transition to $\textcircled{\lhd 0}$ in the reverse direction instead of $\textcircled{\lhd 1}$.

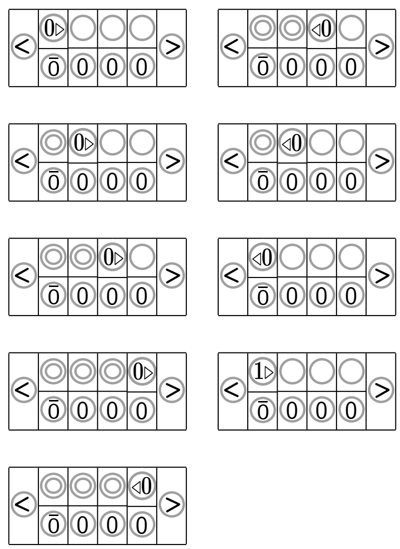
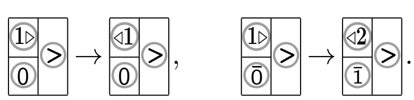


Figure 18: The initial iteration of the Track 1 states on a chain of length six. The figure should be read in column form from top to bottom, left column, then right.

Clock states for the counting phase We illustrate in Figure 19 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.

During the counting phase, Track 2 will always be in some state of the form $\textcircled{<}\textcircled{1}^{*}\textcircled{\bar{0}}\textcircled{0}^{*}\textcircled{>}$, so we will forbid the states $[\textcircled{1\rhd}, \odot]$ and $[\textcircled{\lhd 1}, \odot]$ when $\odot$ is $\textcircled{2}$ or $\textcircled{\bar{1}}$. The right-moving transitions will remain unchanged: $\textcircled{1\rhd}\bigcirc \rightarrow \circledcirc\textcircled{1\rhd}$. 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:



The latter transition triggers the transition to the computation phase. In the left-moving direction, the pointer will sweep past pairs of $\textcircled{0}$ particles and pairs of $\textcircled{1}$ particles:

$$\begin{array}{|c|c|}\hline \circledcirc & \textcircled{\lhd 1} \\ \hline \textcircled{0} & \textcircled{0} \\ \hline \end{array} \to \begin{array}{|c|c|}\hline \textcircled{\lhd 1} & \bigcirc \\ \hline \textcircled{0} & \textcircled{0} \\ \hline \end{array}, \qquad \begin{array}{|c|c|}\hline \circledcirc & \textcircled{\lhd 1} \\ \hline \textcircled{1} & \textcircled{1} \\ \hline \end{array} \to \begin{array}{|c|c|}\hline \textcircled{\lhd 1} & \bigcirc \\ \hline \textcircled{1} & \textcircled{1} \\ \hline \end{array}.$$

When the pointer on Track 1 sweeps left and meets the $\textcircled{\bar{0}}$ particle on Track 2, it triggers an advance of the minute hand. This does not change the transition on Track 1:

$$\begin{array}{|c|c|}\hline \circledcirc & \textcircled{\lhd 1} \\ \hline \textcircled{\bar{0}} & \textcircled{0} \\ \hline \end{array} \to \begin{array}{|c|c|}\hline \textcircled{\lhd 1} & \bigcirc \\ \hline \textcircled{1} & \textcircled{\bar{0}} \\ \hline \end{array}.$$

Note that the $\textcircled{\lhd 1}$ never coincides with the $\textcircled{\bar{0}}$, so we will disallow the state $[\textcircled{\lhd 1}, \textcircled{\bar{0}}]$. Finally, we have that $\textcircled{\lhd 1}$ must turn at the left end:

$$\begin{array}{|c|c|}\hline \textcircled{<} & \textcircled{\lhd 1} \\ \cline{2-2} & \textcircled{1} \\ \hline \end{array} \to \begin{array}{|c|c|}\hline \textcircled{<} & \textcircled{1 \rhd} \\ \cline{2-2} & \textcircled{1} \\ \hline \end{array}.$$

Clock states for the computation phase We illustrate in Figure 20 the first iteration in the computation phase and then the last iteration. The very last state shown is the final state for the clock.

During this time, Track 2 will always be in some state of the form $\textcircled{<}\textcircled{1}^*\textcircled{\bar{1}}\textcircled{2}^*\textcircled{>}$, so we will forbid the states $[\textcircled{2\rhd}, \odot]$ and $[\textcircled{\lhd 2}, \odot]$ when $\odot$ is $\textcircled{0}$ or $\textcircled{\bar{0}}$. Since $\textcircled{2\rhd}$ should never be in the same configuration as $\textcircled{\bar{0}}$, we will also disallow the pair $[\textcircled{2\rhd}, \textcircled{1}][\bigcirc, \textcircled{\bar{0}}]$. The left-moving transitions will remain unchanged: $\circledcirc\textcircled{\lhd 2} \to \textcircled{\lhd 2}\bigcirc$. 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 $\textcircled{1}$ left in Track 2:

$$\begin{array}{|c|c|}\hline \textcircled{<} & \textcircled{\lhd 2} \\ \cline{2-2} & \textcircled{1} \\ \hline \end{array} \to \begin{array}{|c|c|}\hline \textcircled{<} & \textcircled{2 \rhd} \\ \cline{2-2} & \textcircled{1} \\ \hline \end{array}.$$

When the state on Track 2 becomes $\textcircled{<}\textcircled{\bar{1}}\textcircled{2}^*\textcircled{>}$ and the left-moving pointer 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 $\textcircled{<}[\textcircled{\lhd 2}, \textcircled{\bar{1}}]$. In the right-moving direction, the pointer will sweep past pairs of $\textcircled{1}$ particles and pairs of $\textcircled{2}$ particles:

$$\begin{array}{|c|c|}\hline \textcircled{2\rhd} & \bigcirc \\ \hline \textcircled{1} & \textcircled{1} \\ \hline \end{array} \to \begin{array}{|c|c|}\hline \circledcirc & \textcircled{2\rhd} \\ \hline \textcircled{1} & \textcircled{1} \\ \hline \end{array}, \qquad \begin{array}{|c|c|}\hline \textcircled{2\rhd} & \bigcirc \\ \hline \textcircled{2} & \textcircled{2} \\ \hline \end{array} \to \begin{array}{|c|c|}\hline \circledcirc & \textcircled{2\rhd} \\ \hline \textcircled{2} & \textcircled{2} \\ \hline \end{array}.$$

When the pointer on Track 1 sweeps right and meets the $\textcircled{\bar{1}}$ particle on Track 2, it triggers an advance of the minute hand:

$$\begin{array}{|c|c|}\hline \textcircled{2\rhd} & \bigcirc \\ \hline \textcircled{1} & \textcircled{\bar{1}} \\ \hline \end{array} \to \begin{array}{|c|c|}\hline \circledcirc & \textcircled{2\rhd} \\ \hline \textcircled{\bar{1}} & \textcircled{2} \\ \hline \end{array}.$$

Note that since the $\textcircled{2\rhd}$ never coincides with the $\textcircled{\bar{1}}$, we can make the state $[\textcircled{2\rhd}, \textcircled{\bar{1}}]$ illegal. Finally, we have the turning at the right end:

$$\begin{array}{|c|c|}\hline \textcircled{2\rhd} & \textcircled{>} \\ \cline{1-1} \textcircled{2} & \\ \hline \end{array} \to \begin{array}{|c|c|}\hline \textcircled{\lhd 2} & \textcircled{>} \\ \cline{1-1} \textcircled{2} & \\ \hline \end{array}.$$

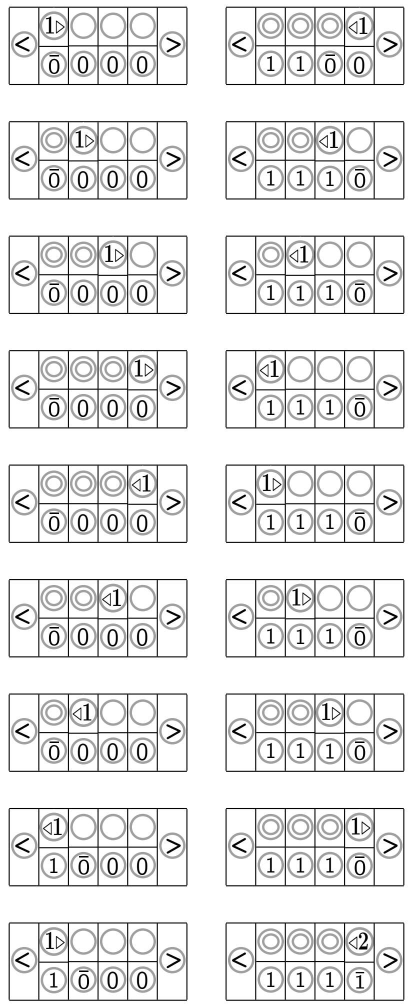


Figure 19: The first iteration (left column) and last iteration (right column) of the counting phase. The very last transition illustrates the transition to the computation phase.

First iteration of the Computation Phase   Last iteration of the Computation Phase

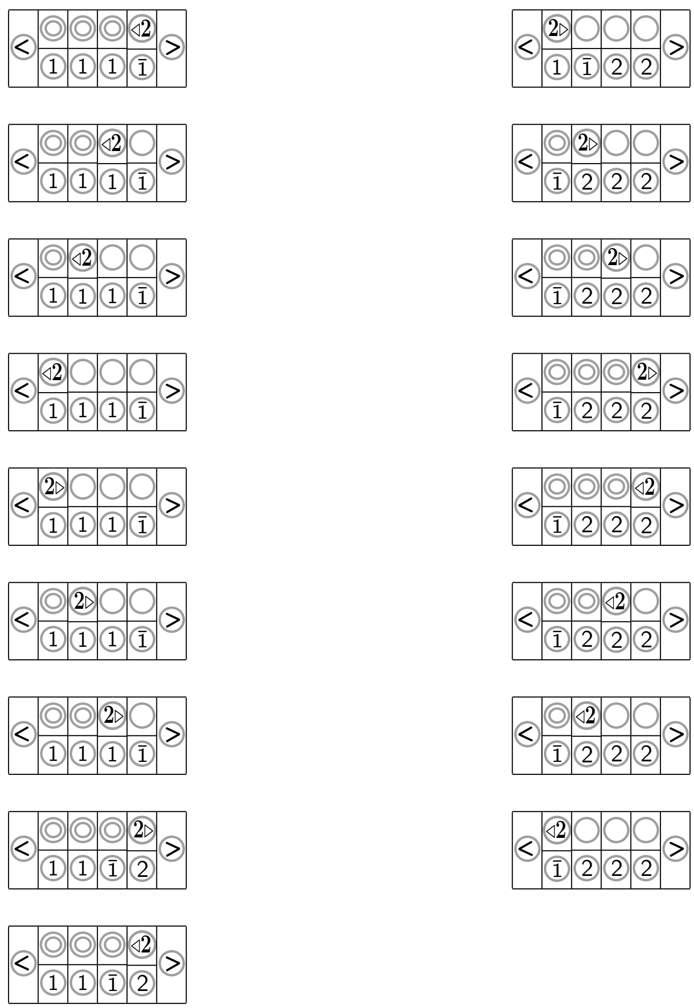


Figure 20: The first and last iterations of the computation phase. The very last state shown is the final state for the clock.

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 $\mathcal{S}_{br}$?

Definition 5.3. We will say that a standard basis state is *well-formed* if the Track 1 state corresponds to a string in the regular language ⧀◎*(▷⃝‖◁⃝)◯*⧁ and if the Track 2 state corresponds to a string in the regular language ⧀①*(⓪̄⓪*‖①̄②*)⧁.

Note that if a standard basis state is well-formed it must be in $\mathcal{S}_{br}$.

Lemma 5.4. *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 15. 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 ⧀⓪▷◯*⧁ on Track 1 and ⧀⓪̄⓪*⧁ on Track 2 and ending with ⧀◁②◯*⧁ on Track 1 and ⧀①̄②*⧁ 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.

Lemma 5.5. *Consider a well-formed standard basis state s that is not in the support of $|\phi_{cl}\rangle$. 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 $|\phi_{cl}\rangle$, 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., [⓪▷, 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 ⧀◎*(⓪▷ + ◁⓪)◯*⧁ occurs in $|\phi_{cl}\rangle$. Moreover, for every standard basis state in the support of $|\phi_{cl}\rangle$, if the control state is ⓪▷ or ◁⓪, then Track 2 is in state ⧀⓪̄⓪*⧁. Now consider a state in which Track 2 has a ①, ①̄ or ② particle and the control particle on Track 1 is ⓪▷ or ◁⓪. The control particle will transition (in either the forward or reverse direction) towards the ①, ①̄ or 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 ①▷ or ◁①. Every possible combination of ⧀◎*(①▷ + ◁①)◯*⧁ occurs in $|\phi_{cl}\rangle$ with every possible combination of ⧀①*⓪̄⓪*⧁ except those where the control particle is ◁① and coincides with the ⓪̄. The state [◁①, ⓪̄] is an illegal particle state. So we now need to take care of the case where the control particle on Track 1 is ①▷ or ◁① and Track 2 is a ⧀①*①̄②*⧁ state. Since it is illegal for a ①▷ or ◁① to coincide with a ①̄ or ② on Track 2, we can assume that the control particle on Track 1 is over a ① particle on Track 2. In this case, it will transition

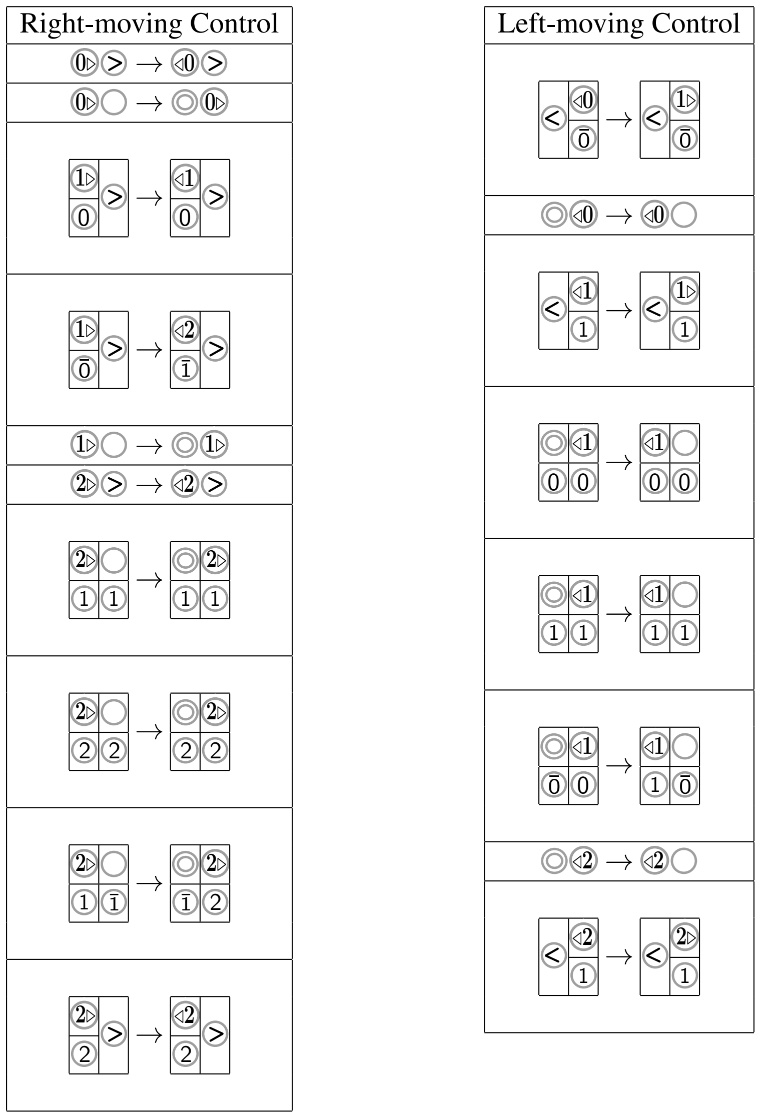


Table 15: The transition rules for Tracks 1 and 2.

in the forward direction in fewer than $2N$ steps (possibly turning at the left end of the chain) until it reaches the ①̄. This will result in a particle in state [①▷, ①̄] which is illegal.

Finally, consider the case where the control particle is ②▷ or ◁②. Every possible combination of the expression ⧀◎*(②▷ + ◁②)◯*⧁ occurs in $|\phi_{cl}\rangle$ with every possible combination of ⧀①*①̄②*⧁ except those where the control particle is ②▷ and coincides with the ①̄. The state [②▷, ①̄] is illegal. So we now need to take care of the case where the control particle on Track 1 is ②▷ or ◁② and Track 2 is a ⧀①*⓪̄⓪*⧁ state. Since it is illegal for a ②▷ or ◁② to coincide with a ⓪̄ or ⓪ on Track 2, we can assume that the control particle on Track 1 is over a 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 ⓪̄ on Track 2. This will result in a pair [②▷, ①][◯, ⓪̄] which is illegal. □

Lemma 5.6. *$|\phi_{cl}\rangle$ is the unique ground state of $H_N|_{\mathcal{S}_{br}}$. 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 [2], so we only give a brief sketch here. The idea is that any standard basis state inside $\mathcal{S}_{br}$ 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 5.4 implies that this graph is composed of disjoint directed paths. Call H_{trans} the Hamiltonian resulting from the sum of all the terms from transition rules and H_{legal} the Hamiltonian from illegal pairs. The subspace spanned by the states in a single path is closed under H_{trans} and H_{legal}. Furthermore, the matrix for H_{trans} restricted to the states in a path looks like

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

where the dimension of the matrix is the length of the path. The matrix for H_{legal} is diagonal in the standard basis. If the path contains no states with illegal pairs, then H_{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 $1/L$, where L is the length of the path. In our case, Lemma 5.5 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 $1/s$. Again, it is known [2] that any state in the subspace spanned by the states in this path will have energy be at least $1/s^3$. By Lemma 5.5, 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 pointer on Track 1 interacts with the other four tracks. Recall that the only role of Track 2 is to control the iterations of Track 1.

5.4 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 ⓪▷ and Track 3 has anything but ◯.

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 ⧀◎*⊙◯*⧁, where ⊙ is a state for M_{BC}. We assume that M_{BC} has a designated start state Ⓢ, so we want to ensure that Track 4 starts out in the configuration ⧀Ⓢ◯*⧁. To do this, we make any state illegal in which the Track 1 state is ⓪▷ and the Track 4 state is anything except Ⓢ or ◯. Furthermore, for any pair in which the left particle is in state ⧀, and the Track 1 state is ⓪▷, the Track 4 state must be Ⓢ. For any pair in which the left particle is not in state ⧀, and the Track 1 state is ⓪▷, the Track 4 state must be ◯. 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.

5.5 Counting phase

Every time the 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 ②▷ symbol is what causes the Turing machine V to execute a step.

5.6 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 ②▷ particle triggers the execution of a step, we will consider particles as triplet states of the form [②▷$,q,a]$, [◯$,q,a]$ and [◎$,q,a]$, for $a \in \Sigma$ and $q \in Q \cup \{$◯$\} \cup \{$◎$\}$. 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 ◯ or ◎. We will also be interested in the *computation state* of a particle which just consists of a pair $[q,a]$, where $q \in Q \cup \{$◯$\} \cup \{$◎$\}$ 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 [5] 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 ②▷ state on Track 1 in the same location. It will also be forbidden to have the ②▷ 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 $q,p \in Q \cup \{◯\} \cup \{◎\}$. $\mathcal{S}_A$ is the space spanned by all such legal basis states such that $q \notin Q_L$, $p \notin Q'_R$. Note that if the Track 1 state for a pair of particles is $|\text{②▷◯}\rangle$, then the only possible computation states for the pair are in $\mathcal{S}_A$. We also define $\mathcal{S}_B$ to be the space spanned by all legal basis states of the form $[q,a][p,b]$ such that $p \notin Q_L$, $q \notin Q'_R$. Similarly, if the Track 1 state for a pair of particles is $|\text{◎②▷}\rangle$, then the only possible legal computation states for the pair are in $\mathcal{S}_B$.

We will describe a transformation T that maps $\mathcal{S}_A$ to $\mathcal{S}_B$. The extended map $T \otimes |\text{◎②▷}\rangle\langle\text{②▷◯}|$ 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 $\mathcal{S}_A$ works in two parts. At the moment that the Track 1 pointer 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 ②▷ 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.

$$\left|\begin{array}{|c|c|}\hline \text{②▷} & \text{◯} \\ \hline \text{◎} & q \\ \hline \text{⊙} & a \\ \hline\end{array}\right\rangle \to \sum_{a,b,q_L,q} \delta(q,a,q_L,b,L) \left|\begin{array}{|c|c|}\hline \text{◎} & \text{②▷} \\ \hline q_L & \text{◯} \\ \hline \text{⊙} & b \\ \hline\end{array}\right\rangle + \sum_{a,b,q_R,q} \delta(q,a,q_R,b,R) \left|\begin{array}{|c|c|}\hline \text{◎} & \text{②▷} \\ \hline \text{◎} & q'_R \\ \hline \text{⊙} & b \\ \hline\end{array}\right\rangle. \tag{5.1}$$

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 ②▷ 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 ②▷, that triggers the execution of the right-moving step.

$$\left|\begin{array}{|c|c|}\hline \text{②▷} & \text{◯} \\ \hline q'_R & \text{◯} \\ \hline \text{⊙} & \text{⊙} \\ \hline\end{array}\right\rangle \to \left|\begin{array}{|c|c|}\hline \text{◎} & \text{②▷} \\ \hline \text{◎} & q_R \\ \hline \text{⊙} & \text{⊙} \\ \hline\end{array}\right\rangle. \tag{5.2}$$

Otherwise, the ②▷ state just sweeps to the right, leaving the other tracks unchanged:

$$\left|\begin{array}{|c|c|}\hline \text{②▷} & \text{◯} \\ \hline q_R & \text{◯} \\ \hline \text{⊙} & \text{⊙} \\ \hline\end{array}\right\rangle \to \left|\begin{array}{|c|c|}\hline \text{◎} & \text{②▷} \\ \hline q_R & \text{◯} \\ \hline \text{⊙} & \text{⊙} \\ \hline\end{array}\right\rangle, \quad \left|\begin{array}{|c|c|}\hline \text{②▷} & \text{◯} \\ \hline \text{◯} & \text{◯} \\ \hline \text{⊙} & \text{⊙} \\ \hline\end{array}\right\rangle \to \left|\begin{array}{|c|c|}\hline \text{◎} & \text{②▷} \\ \hline \text{◯} & \text{◯} \\ \hline \text{⊙} & \text{⊙} \\ \hline\end{array}\right\rangle, \quad \left|\begin{array}{|c|c|}\hline \text{②▷} & \text{◯} \\ \hline \text{◎} & \text{◎} \\ \hline \text{⊙} & \text{⊙} \\ \hline\end{array}\right\rangle \to \left|\begin{array}{|c|c|}\hline \text{◎} & \text{②▷} \\ \hline \text{◎} & \text{◎} \\ \hline \text{⊙} & \text{⊙} \\ \hline\end{array}\right\rangle. \tag{5.3}$$

Note that transformations (5.1), (5.2) and (5.3) define T on every state in $\mathcal{S}_A$. Furthermore, the image of $T|_{\mathcal{S}_A}$ is in $\mathcal{S}_B$. A critical fact is that after T is applied to a pair, the computation state of that pair is in $\mathcal{S}_B$, which implies that the next pair over is now in subspace $\mathcal{S}_A$ 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.

Claim 5.7. *T is a partial isometry (meaning it preserves inner products) when restricted to $\mathcal{S}_A$.*

Since $\mathcal{S}_A$ and $\mathcal{S}_B$ have the same dimension, the claim implies that $T^\dagger$ is well-defined and a partial isometry on $\mathcal{S}_B$. We can extend T to be a unitary map on the full Hilbert space. Then we select the Hamiltonian term to be:

$$I_{\mathcal{S}_A} \otimes |\textcircled{2}\!\rhd \bigcirc\rangle\langle \textcircled{2}\!\rhd \bigcirc| + I_{\mathcal{S}_B} \otimes |\circledcirc \textcircled{2}\!\rhd\rangle\langle \circledcirc \textcircled{2}\!\rhd| - T \otimes |\circledcirc \textcircled{2}\!\rhd\rangle\langle \textcircled{2}\!\rhd \bigcirc| - T^\dagger \otimes |\textcircled{2}\!\rhd \bigcirc\rangle\langle \circledcirc \textcircled{2}\!\rhd| .$$

Proof of claim. Expressions (5.1), (5.2) and (5.3) each define T on a different orthogonal subspace of $\mathcal{S}_A$. $\mathcal{S}_{A1}$ is the space of the form $[\circledcirc, \odot][q,a]$, $\mathcal{S}_{A2}$ is the space of states of the form $[q'_R, \odot][\bigcirc, \odot]$ and $\mathcal{S}_{A3}$ is the space spanned by states of the form $[q_R, \odot][\bigcirc, \odot]$, $[\bigcirc, \odot][\bigcirc, \odot]$ and $[\circledcirc, \odot][\circledcirc, \odot]$. We will argue that T is a partial isometry when restricted to each of $\mathcal{S}_{A1}$, $\mathcal{S}_{A2}$ and $\mathcal{S}_{A3}$. Furthermore the images of $T_{\mathcal{S}_{A1}}$, $T_{\mathcal{S}_{A2}}$ and $T_{\mathcal{S}_{A3}}$ are mutually orthogonal. This implies that T is a partial isometry on $\mathcal{S}_A$.

T is clearly a partial isometry on each of $\mathcal{S}_{A2}$ and $\mathcal{S}_{A3}$. We only need to examine the transformation given in (5.1). 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 (5.1) (T restricted to states of the form $[\circledcirc, \odot][q,a]$) is a partial isometry. □

By construction, a single iteration will implement the correct transition rule for the TM head in a single specific position. We should also verify that it acts correctly when the head is in a superposition of 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 5.8. 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 $\textcircled{1}\!\rhd$. 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 $\textcircled{1}\!\rhd$. 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 $\textcircled{2}\!\rhd$. 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 $\textcircled{2}\!\rhd$. 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. We can add illegal pairs so that any invalid state has an energy penalty of at least one. 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.

5.7 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 5.3 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. It will be convenient to separate the states of Tracks 3 through 5 from the state of Track 6. So let X be a standard basis state for Tracks 3 through 5 and Y be a standard basis state for Track 6. Let $|t\rangle$ be the state of the clock after t transitions. Define $|\phi_{t,X,Y}\rangle$ 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 $|\phi_{X,Y}\rangle = \sum_{t=0}^{T-1} |t\rangle|\phi_{t,X,Y}\rangle$. The $|\phi_{X,Y}\rangle$ 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, $\bigcirc^*$ on Track 3 and $Ⓢ\bigcirc^*$ and $q_0\bigcirc^*$ on Tracks 4 and 5. The energy for any $|\phi_{X,Y}\rangle$ where $X \neq I$ will be at least $\Omega(1/N^2)$. This is because if $X \neq I$, at least one state in the initialization phase will have an energy penalty as the $0\!\triangleright$ control state sweeps to the right. So the subspace spanned by $\{|\phi_{I,Y}\rangle\}$ for all Y form a basis of the zero eigenspace for the Hamiltonian terms defined so far, restricted to $\mathcal{S}_{br}$, the subspace of all bracketed states. The next highest eigenstate in $\mathcal{S}_{br}$ has energy $\Omega(1/N^3)$.

5.8 Accepting states

Now we add a term for those $|\phi_{I,\psi}\rangle$ that do not lead to an accepting computation for V. In the last configuration of the clock, the two left-most particles are in state $[<][\triangleleft 2, \bar{1}]$. We assume that an accepting computation will end with the head in the left-most place in state q_A.

We add a penalty for a pair whose Track 1 and 2 state is $[<][\triangleleft 2, \bar{1}]$ 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 ε/N^2 for this term where ε can be made arbitrarily small. If no witness leads to an accepting computation, all states will have energy at least $(1-\varepsilon)/N^2$ from this term.

5.9 Boundary conditions

To complete the proof of Theorem 2.5, proving the hardness of 1-DIM TIH on a finite chain, we add a term that will enforce that the ground state is in $\mathcal{S}_{br}$. In order to penalize states that are not bracketed, we weight H by a factor of 3 and add in the term $I - |\textcircled{>}\rangle\langle\textcircled{>}| - |\textcircled{<}\rangle\langle\textcircled{<}|$ to every particle. This has the effect of adding energy $N-2$ to every state in $\mathcal{S}_{br}$. The ground space is still spanned by the $|\phi_{I,Y}\rangle$ and the spectral gap remains $\Omega(1/N^3)$. Any standard basis state outside $\mathcal{S}_{br}$ will have a cost of at least $N-1$: If there are a particles in state $\textcircled{>}$ or $\textcircled{<}$ in the middle of the chain, they will save a from the $I - |\textcircled{>}\rangle\langle\textcircled{>}| - |\textcircled{<}\rangle\langle\textcircled{<}|$ term. However, there will be at least $\lceil a/2 \rceil$ illegal pairs because the $\textcircled{<}$ particles in the middle will each form an illegal pair with the particle to their left and the $\textcircled{>}$ 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 ($\textcircled{<}$ or $\textcircled{>}$) which is more plentiful, yielding at least $\lceil a/2 \rceil$ 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.

6 The quantum cycle

If we instead have periodic boundary conditions, the problem is still $\mathsf{QMA}_{\mathsf{EXP}}$-complete. The only part in the construction in Section 5 that needs to change are the boundary conditions described in Section 5.9. We start by removing the penalty for the pair $\textcircled{>}\textcircled{<}$. 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:

$$\underbrace{\textcircled{<}\ \cdots\ \textcircled{>}}_{\text{Segment 1}}\underbrace{\textcircled{<}\ \cdots\ \textcircled{>}}_{\text{Segment 2}}\underbrace{\textcircled{<}\ \cdots\ \textcircled{>}}_{\text{Segment 3}}.$$

Note that the transition rules do not change the locations or numbers of $\textcircled{<}$ or $\textcircled{>}$ 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 $\textcircled{>}$ has a $\textcircled{<}$ to its immediate right and every occurrence of $\textcircled{<}$ has a $\textcircled{>}$ it 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.

We will use only odd N and add a 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 Ⓐ, Ⓑ, and ⦶. Even the ⧁ or ⧀ particles have a Track 0 state, so the state of a particle is either in $\{$Ⓐ, Ⓑ, ⦶$\} \times \{$⧀, ⧁$\}$ or it is a 7-tuple denoting states for Tracks 0 through 6.

We add the terms $|$ⒶⒶ$\rangle\langle$ⒶⒶ$|$ and $|$ⒷⒷ$\rangle\langle$ⒷⒷ$|$, which gives an energy penalty of 1 for pairs ⒶⒶ or ⒷⒷ 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 ⦶. Now we add an energy penalty of 1 for any particle whose Track 0 state is ⦶ which is not [⦶, ⧀]. We also add an energy penalty for [Ⓐ, ⧀] or [Ⓑ, ⧀]. The only way for a state to have energy less than one is for there to be at least one segment and to have a ⦶ on Track 0 exactly at the locations where there is a ⧀ on the other tracks. Finally, we add an energy penalty of $1/2$ for any particle whose Track 0 state is ⦶. 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 ⦶ on Track 0 exactly at the single location where there is a ⧀ on the other tracks.

Consider the set of all basis states with exactly one segment and whose Track 0 state has a ⦶ co-located with the ⧀ site and alternating Ⓐ's and Ⓑ'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 ⦶ site and the string of alternating Ⓐ's and Ⓑ's can begin with Ⓐ or Ⓑ. 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 $1/2 + \Omega(1/N^3)$.

7 Quantum case with reflection symmetry

Many of the Hamiltonian terms in Section 5 had a left-right asymmetry, allowing us to, for instance, start the Turing machine head at the left end of the line of particles. When we add reflection symmetry to the translational invariance, this is no longer as straightforward. However, by increasing the number of states and with an appropriate choice of Hamiltonian, we will be able to spontaneously break the reflection symmetry in the vicinity of the pointer states, showing that 1-DIM TIH with reflection symmetry is $\mathsf{QMA}_{\mathsf{EXP}}$-complete. In this section, we present the construction for periodic boundary conditions, but a similar construction works with open boundary conditions.

The basic idea of the construction is to force the pointer states to have Ⓐ on one side and Ⓑ on the other. That creates an asymmetry between the two directions, which we can use to define left transitions and right transitions. At one time step, we should consider Ⓐ to be on the left and Ⓑ to be on the right. However, in the next time step, the pointer has moved, and now Ⓑ should be on the left and Ⓐ should be on the right. Therefore, we will split the pointer state into pairs of states that keep track of which direction Ⓐ should be in.

Now for the details of the construction. We assume that N, the number of particles in the cycle, is odd. There will be a total of seven tracks, numbered 0 through 6. A state is then specified either as ⦶ or by a 7-tuple, specifying the state for each Track. The state ⦶ spans all seven tracks. Some of the energy penalties we introduce will be a multiple of some constant c to be chosen later.

The states for each of the Tracks 1, 2, 4, and 5 will be divided into control states and non-control states. The non-control states will be divided into two types: A-states and B-states.

Definition 7.1. For any particular standard basis state, we say that its *configuration* is defined by the following properties:

1. the number of control particles on Tracks 1, 2, 4, and 5
2. the sequence of A-states and B-states (with control states removed) on Tracks 1, 2, 4, and 5
3. the entire state of Track 0
4. the locations of the particles in state ⦶.

We will select the transition rules so that they never change the configuration of a standard basis state. Thus, if we consider the subspace spanned by all the standard basis states with a particular configuration, the Hamiltonian will be closed on that subspace. We can therefore analyze each such subspace separately. We will specifically be interested in the following properties:

Claim 7.2. *With an appropriate choice of Hamiltonians, we can ensure that the ground state is a superposition of basis states whose configurations satisfy the following properties:*

1. *There is one particle in* ⦶.
2. *For Track 0, the rest of the particles alternate* Ⓐ *and* Ⓑ.
3. *Tracks 1, 2, 4, and 5 each have exactly one control particle.*
4. *The non-control particles for tracks 1, 2, 4, and 5 alternate between A-states and B-states with an A-state or control state on either side of the* ⦶.
5. *The control particle is flanked by an A-state on one side and a B-state or a* ⦶ *on the other.*

We will therefore restrict our attention to configurations that have these properties.

Proof of claim. Towards this end, we add the following constraints: Track 0 has states Ⓐ, Ⓑ and ⦶. There is a penalty of $2c$ for the pairs ⒶⒶ and ⒷⒷ. There is a penalty of c for any occurrence of ⦶.

Any state that has no ⦶ or more than one ⦶ will have a cost of at least $2c$. A configuration which satisfies properties 1 and 2 will have a cost of c.

For Tracks 1, 2, 4, and 5, we will enforce that there is exactly one particle that is in a control state. This control state can move around, but there should be just one. This is enforced as follows. The control states for a track all cost 1. There will be two states for each non-control state, an A version and a B version. The state ⦶ spanning Tracks 0 through 6 must go next to an A-state or a control state on Tracks 1, 2, 4, and 5. The A-states cannot be next to each other and the B-states cannot be next to each other. The control states can be next to the ⦶, A-states, or B-states, but not next to each other. All of these forbidden pairs have a cost of c. These rules are summarized in Table 16. The transitions may cause the control particle to shift over by one site by swapping places with a non-control particle. In these transitions, the non-control particle will remain the same with respect to whether it is an A or B state although it may change other aspects of its state.

	$\textcircled{\vert}$	A	B	control
$\textcircled{\vert}$	c	0	c	0
A	0	c	0	0
B	c	0	c	0
control	0	0	0	c

Table 16: Summary of the rules for A and B-states in Tracks 1, 2, 4, 5. Also, all control states have an additional cost of 1.

Property 4 of the claim follows immediately from these constraints.

If there are no control states on a Track, the total cost of the Track must be at least c (in addition to the c cost of having the one $\textcircled{\vert}$). This is a consequence of odd N—alternating A-states and B-states would then produce an A-state on one side of $\textcircled{\vert}$, but a B-state on the other side.

Any configuration that avoids the cost of c must have at least one control state on each of Tracks 1, 2, 4, and 5. Each control state incurs a cost of one. Thus, the minimal-cost configuration has exactly one control state on each track of 1, 2, 4, and 5 with a total cost of $c+4$. Any other configuration will have a cost of at least $2c$. The minimum cost configuration will have a control particle somewhere within a sequence of alternating B's and A's with an A on each end next to the $\textcircled{\vert}$. In order to maintain the parity, property 5 must hold.

We can choose $c = 5$ to guarantee that there is an energy gap of at least one between configurations which obey properties 1 through 5 in the claim and those which do not. □

7.1 Track 1 transitions

We are now ready to describe the transitions for Track 1. Since these rules satisfy reflective symmetry, whenever there is a rule: $ab \to cd$, there is also a rule $ba \to dc$. For the sake of brevity, we do not always give the reflected version of the rule with the understanding that it is implicit.

There are four types of control states: $\textcircled{A\triangleright}$, $\textcircled{B\triangleright}$, $\textcircled{\triangleleft A}$ and $\textcircled{\triangleleft B}$. When we discuss Track 2, these will each be expanded into three varieties as was done in the construction without reflection symmetry. There are only two non-control states. These are labelled $\textcircled{A}$ and $\textcircled{B}$. We have the following transitions:

$$\textcircled{A\triangleright}\textcircled{A} \to \textcircled{A}\textcircled{B\triangleright}, \qquad \textcircled{A}\textcircled{\triangleleft A} \to \textcircled{\triangleleft B}\textcircled{A}, \qquad \textcircled{B\triangleright}\textcircled{B} \to \textcircled{B}\textcircled{A\triangleright}, \qquad \textcircled{B}\textcircled{\triangleleft B} \to \textcircled{\triangleleft A}\textcircled{B}.$$

That is, the control state swaps its location with an adjacent non-control state, and the pointer stays pointing the same direction while the A/B label on the pointer switches.

We have the following transitions at the ends

$$\textcircled{\triangleleft B}\textcircled{\vert} \to \textcircled{A\triangleright}\textcircled{\vert}, \qquad \textcircled{B\triangleright}\textcircled{\vert} \to \textcircled{\triangleleft A}\textcircled{\vert}.$$

That is, the pointer switches direction, and the A/B label on the pointer switches as well.

It will be useful to fix a direction for the cycle by disconnecting the cycle at the $\textcircled{\vert}$ particle and stretching out from left to right.

Definition 7.3. Once this direction is fixed, we say that a standard basis state is *correctly oriented* if one of the properties hold:

Control particle is (A▷) and the particle to its right is in state (A).

Control particle is (◁A) and the particle to its left is in state (A).

Control particle is (B▷) and the particle to its right is in state (B) or (|).

Control particle is (◁B) and the particle to its left is in state (B) or (|).

In other words, the pointer points to a state which matches the A/B label on the pointer.

There are $2(n-1)$ standard basis states for Track 1 that satisfy properties 1 through 4 of Claim 7.2 which are also correctly oriented. The reflection of each of these states is a standard basis state which satisfies properties 1 through 4 but is not correctly oriented. The transitions form an infinite loop over the correctly oriented Track 1 states as shown below. The (|) particle is duplicated on each end to illustrate the transitions.

(|)(A▷)(A)(B)··· (B)(A)(|)	(|)(A)(B)··· (B)(A)(◁A)(|)
(|)(A)(B▷)(B)··· (B)(A)(|)	(|)(A)(B)··· (B)(◁B)(A)(|)
(|)(A)(B)(A▷)··· (B)(A)(|)	(|)(A)(B)··· (◁A)(B)(A)(|)
···	···
(|)(A)(B)··· (B▷)(B)(A)(|)	(|)(A)(B)(◁B)··· (B)(A)(|)
(|)(A)(B)··· (B)(A▷)(A)(|)	(|)(A)(◁A)(B)··· (B)(A)(|)
(|)(A)(B)··· (B)(A)(B▷)(|)	(|)(◁B)(A)(B)··· (B)(A)(|)

Note that we always have a two-fold degeneracy which results from reflecting each state in this sequence. This results in a set of states which are not correctly oriented. The transitions are closed on the two sets of states. Furthermore, the structure of the transitions is identical on each set, so we can select one orientation and analyze that subspace. For ease of exposition then we will restrict our attention to states on Track 1 which are correctly oriented.

We no longer need distinct endmarkers (>) and (<) since the state of the control particle next to an endmarker indicates whether it will sweeping to the other end or changing directions first. For example any transition or illegal pair involving the pair (<)(◁) would be replaced by (|)(◁B). The pair (◁B)(|) will never occur in a state which is correctly oriented. The table below shows the pairs involving end states in the non-reflective construction and the corresponding pairs in the reflecting construction that we are describing in this section.

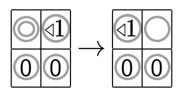
(<)(◁)	(|)(◁B), (◁B)(|)
(<)(▷)	(|)(A▷), (A▷)(|)
(◁)(>)	(|)(◁A), (◁A)(|)
(▷)(<)	(|)(B▷), (B▷)(|)

In the non-reflective construction, the Track 1 control symbol (◁), has three varieties (◁0), (◁1) and (◁2). Now in the construction with reflection symmetry, these in turn come in an A-version and B-version giving six control symbols. In order to keep the notation from becoming too dense, we denote the A-version with a heavier outline as in **(◁0)**, **(◁1)**, **(◁2)** (these are the three varieties of (◁A)) and the B-version will be denoted by a lighter outline as in (◁0), (◁1), (◁2) (these are the three varieties of (◁B)). We have a similar set of the right-pointing control states.

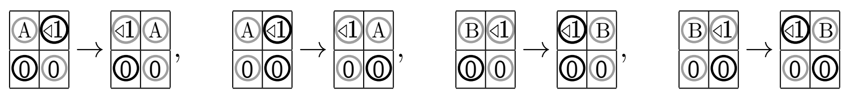
7.2 Track 2

As discussed above, we will enforce that Track 2 will have exactly one particle in a control state. For Track 2, the control states are $\bar{0}$ and $\bar{1}$. The other states 0, 1 and 2 will each have an A-version and a B-version. We again use the notational convention that the A-version will be denoted by a heavier outline as in 0 and the B-version will be denoted by a lighter outline as in 0. In addition to the constraints described below, the states satisfy the constraints given in Table 16.

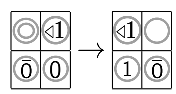
We need to have a transition for every possible combination of A's and B's. Thus the transition:



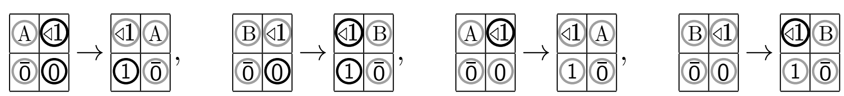
is replaced by four transitions:



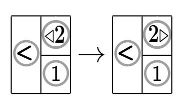
Similarly, the transition



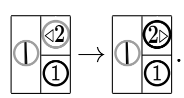
is replaced by



Finally, the transitions involving an end particle will be expressed using the appropriate control character as expression in the table above. The rule



will become



Although all of the illegal pairs and transitions have corresponding rules obtained by reflection, only one will apply since the states are correctly oriented and the direction of the Track 0 control particle is determined.

In the construction without reflection symmetry, we defined a Track 2 state to be well-formed if it had the following form: $< 1^* (\bar{0} 0^* \| \bar{1} 2^*) >$. With the new construction, we use the following definition instead:

Definition 7.4. A Track 2 state is *well-formed* if it has the form:

$$< (1\,1)^* (\bar{0} (0\,0)^* 0 \| \bar{1} (2\,2)^* 2) > \quad \text{or} \quad < (1\,1)^* 1 (\bar{0} (0\,0)^* \| \bar{1} (2\,2)^*) > .$$

The set of well-formed basis states are all closed under the transition rules. If we restrict our attention to Tracks 1 and 2 and the subspace spanned by all well-formed clock states, the results from the previous construction apply and we know that the unique ground state is the uniform superposition over valid clock states.

The construction without reflection symmetry makes use of illegal pairs with a particular orientation in enforcing that states must be well-formed. We need a means of enforcing this same property with only illegal pairs that obey reflection symmetry. Since we have restricted our attention to states with only one control particle, we know that the Track 2 state must have the form $\textcircled{|}(\textcircled{0}\|\textcircled{1}\|\textcircled{2})^*(\textcircled{\bar{0}}\|\textcircled{\bar{1}})(\textcircled{0}\|\textcircled{1}\|\textcircled{2})^*\textcircled{|}$. We will call a state of this form *proper*. Furthermore, the set of transition rules is closed over the subspace spanned by this set.

We will first observe that for any proper standard basis state, at most one transition rule applies in the forward direction and at most one applies in the reverse direction. This follows from the fact that the control particle on Track 1 and the location of the A-state and B-state on either side of the control particle uniquely determine which pair the transition will apply to. Then the argument for the previous construction (Lemma 4.4) carries over to establish that the transitions that apply in each direction are unique.

This means that when we look at the state graph (graph of all standard basis states with a directed edge for each transition), it forms a set of disjoint paths over all proper standard basis states. Since the transition rules are closed over well-formed states, we know that a path is either composed entirely of states that are well-formed or states that are not well-formed. Then we will add a set of illegal pairs which all have the desired reflection symmetry and show that for every path that is not composed of well-formed states, a fraction of $1/2n$ of the states must have a state with an illegal pair. This will imply that any state in the subspace formed by the closure of standard basis states long a path composed of states that are not well-formed will have energy $1/poly(n)$ more than the uniform superposition of the valid clock states.

We add a set of illegal pairs, each with a cost of one. For each pair specified, its reflection is also illegal. This first group are $\textcircled{0}\textcircled{1}$, $\textcircled{0}\textcircled{2}$, $\textcircled{1}\textcircled{2}$. Each of these states have an A-version and a B-version and we implicitly disallow any combination of these. That is the pair $\textcircled{0}\textcircled{1}$ implicitly denotes the two pairs $\textcircled{\mathbf{0}}\textcircled{1}$ and $\textcircled{0}\textcircled{\mathbf{1}}$. Recall that it is already illegal to have two A-states or two B-states next to each other. Furthermore, the control states can not be next to each other, so $\textcircled{\bar{1}}\textcircled{\bar{0}}$ is an illegal pair. These rules enforce that any state without an illegal pair must be of the form:

$$\textcircled{|}(\textcircled{0}^*\|\textcircled{1}^*\|\textcircled{2}^*)(\textcircled{\bar{0}}\|\textcircled{\bar{1}})(\textcircled{0}^*\|\textcircled{1}^*\|\textcircled{2}^*)\,.$$

We also add pairs $\textcircled{\bar{0}}\textcircled{2}$ and $\textcircled{\bar{1}}\textcircled{0}$ since these never appear in a well-formed state. Finally, we want to have the $\textcircled{1}^*$ on the left end of the chain and the $\textcircled{0}^*$ or $\textcircled{2}^*$ on the right end of the chain. To do this, we add illegal pairs:

$$\begin{array}{|c|c|}\hline \textcircled{|} & \textcircled{\triangleleft B} \\ \cline{2-2} & \textcircled{0} \\ \hline \end{array}\,, \qquad \begin{array}{|c|c|}\hline \textcircled{|} & \textcircled{\triangleleft B} \\ \cline{2-2} & \textcircled{2} \\ \hline \end{array}\,, \qquad \text{and} \qquad \begin{array}{|c|c|}\hline \textcircled{B \triangleright} & \textcircled{|} \\ \cline{1-1} \textcircled{1} & \\ \hline \end{array}\,..$$

We also add illegal pairs:

$$\begin{array}{|c|c|}\hline \textcircled{|} & \textcircled{A \triangleright} \\ \cline{2-2} & \textcircled{0} \\ \hline \end{array}\,, \qquad \begin{array}{|c|c|}\hline \textcircled{|} & \textcircled{A \triangleright} \\ \cline{2-2} & \textcircled{2} \\ \hline \end{array}\,, \qquad \text{and} \qquad \begin{array}{|c|c|}\hline \textcircled{\triangleleft A} & \textcircled{|} \\ \cline{1-1} \textcircled{1} & \\ \hline \end{array}\,.$$

These are duplicated for the A-version and B-version of $\textcircled{0}$, $\textcircled{1}$, and $\textcircled{2}$ as well as $\textcircled{\triangleleft 0}$, $\textcircled{\triangleleft 1}$, and $\textcircled{\triangleleft 2}$ for $\textcircled{\triangleleft B}$ and $\textcircled{\triangleleft \mathbf{0}}$, $\textcircled{\triangleleft \mathbf{1}}$, and $\textcircled{\triangleleft \mathbf{2}}$ for $\textcircled{\triangleleft A}$. If we have a proper state that is not well-formed and has no illegal pair, it must either have a $\textcircled{1}^*$ on the right end of the chain or a $\textcircled{0}^*$ or $\textcircled{2}^*$ on the left end of the chain. What will happen, is that the control symbol on Track 1 will cycle around to the end of the chain where the violation occurs and reach a state with an illegal pair.

7.3 Turing machine initialization

We have already described a way to enforce that there is one control state on Tracks 4 and 5. The particle states which correspond to Turing Machine states will be the control states for Tracks 4 and 5. There is some designated start state $\textcircled{S}$ for each Turing Machine and we want to ensure that the Turing Machine starts with the head in the leftmost location in state $\textcircled{S}$. Similar to the non-reflection case, we make any state illegal in which the Track 1 state is $\textcircled{\mathbf{0}\triangleright}$ or $\textcircled{0\triangleright}$ and the Track 4 state is anything except $\textcircled{S}$ or $\bigcirc$. As described above, we have already enforced that there is one state corresponding to the head location. Finally, we make any pair illegal in which one particle is in state $\textcircled{|}$ and the other particle has a Track 1 state of $\textcircled{\mathbf{0}\triangleright}$ and the Track 4 state is not $\textcircled{S}$. This ensures that the unique head location is at the left end of the chain. The same is done for Track 5.

7.4 Turing machine implementation

All the transitions are expended as for Track 2 to include the A-version and the B-version of the non-control states. Transitions involving $\textcircled{>}$ and $\textcircled{<}$ are replaced by rules which use $\textcircled{|}$ and the appropriate control particle for Track 1 which will determine whether the other particle is to the left or the right of the $\textcircled{|}$.

All the transition rules for the execution of the Turing Machines are governed by the direction in which the $\textcircled{\triangleleft A}$ and $\textcircled{\triangleleft B}$ control particles move. We can add the reflection of each of these rules and since the direction of the control particle is fixed, only the correct set of rules will apply.

8 The infinite chain

One common approach to studying translationally-invariant systems in physics is to take the limit of the number of particles going to ∞. In this case, we have to alter our approach slightly in order to define a sensible notion of complexity. For one thing, we have to change the quantity we wish to evaluate. In general, the ground state energy will be infinite when there are infinitely many particles. Instead, we should study the energy density in the ground state: the energy per particle.

There is an additional complication. Throughout the rest of the paper, we have fixed a single Hamiltonian term and used it for all values of N, but if we were to do that for the infinite chain, the desired answer (the ground state energy) would be just a single number, and there would be no language for us to study the complexity of.

The solution is to vary the Hamiltonian term. We still constrain it to interact two particles (i. e., act on a d^2-dimensional Hilbert space), but now we let the entries of the matrix vary. In particular, if we specify terms in the Hamiltonian to $n = \Theta(\log N)$ bits of precision, we can cause the ground state of the infinite chain to break up into segments each of size N. Within a single segment, the Hamiltonian will act like the usual 1D Hamiltonian discussed in Section 5. The reader is referred back to the formal definition of the problem, Definition 2.6 in Section 2.

One might be concerned that the problem is not well-defined on an infinite chain. After all, the hardness constructions used in this paper cause the ground state to change considerably as N varies, suggesting that there is no well-defined limit as $N \to \infty$. This concern is largely addressed by separating N and the number of particles, since the ground state is now the tensor product of segments of finite size, each of which is completely well-defined. To be even more careful, one can take a finite number of particles $M \gg N$, using either periodic or open boundary conditions. If we write down the restriction to M particles of the infinite ground state considered below, there is a constant-size correction to the energy due to the boundary conditions. However, we wish to determine the energy *per particle*, and the correction to that is only $O(1/M)$. Thus, in the limit $M \to \infty$, the energy per particle of the ground state is well-defined.

The first step is to alter the Hamiltonian for an N-particle chain slightly so that we can assume that if a chain of length N has a zero energy state in $\mathcal{S}_{br}$, then it must be the case that $N = 3 \bmod 4$. This can be achieved by having an additional track with states A, B, C and D. We will use illegal pairs to enforce that this additional track has states that are a substring of $(ABCD)^*$. Then we add illegal pairs to enforce that only an A can be adjacent to a ⧀ or a ⧁. Therefore, in order to avoid additional cost, the additional track must be of the form ⧀$A(BCDA)^*$⧁. In order to achieve the same expressive power as we originally had, the verifier Turing Machine V will begin by erasing the two least significant bits of its input (the count from M_{BC}). Thus, the new construction will yield a state with energy ε on a chain of length N if and only if $N = 3 \bmod 4$ and there is a quantum witness which causes the verifier to accept a binary representation of $(N-3)/4$ with probability $(1-\varepsilon)$ in $N-2-x$ steps, where x is the number of steps required for the subroutine which deletes the last two bits of the input. x is approximately $2\log N$. If either $N \neq 3 \bmod 4$ or every witness causes V to reject with high probability on input $(N-3)/4$, the ground state energy will be at least $\Omega(1/N^3)$. For the remainder of this section, we will assume that $N = 3 \bmod 4$.

Now we address the case of the infinite chain in earnest. For the next step, we make a second small change to the set of illegal pairs to allow the pair ⧁⧀.

Definition 8.1. For a particular state, the sequence of sites extending from a ⧀ site through the next ⧁ site is a *segment*.

The set of bracketed 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 along the chain. If a standard basis state contains no illegal pairs, there must be a control symbol on Tracks 1 and 2 between ⧀ and ⧁ which means we cannot have segments of length 2. Also, every occurrence of ⧁ has a ⧀ to its immediate right and every occurrence of ⧀ has a ⧁ to its immediate left. In addition, the illegal pairs exclude the possibility of sequences of sites containing two ⧀ sites with no ⧁ in between, or sequences with two ⧁ sites without a ⧀. We therefore get the following lemma:

Lemma 8.2. *A standard basis state with no illegal pairs either contains no ⧀ or ⧁ sites anywhere in the chain, or it can be divided up into valid segments with no extra sites in between segments.*

Let H_{TM} be the resulting two-particle term that includes illegal pairs and transition rules. We omit for now the term which penalizes for a rejecting computation. A state thus has zero energy for H_{TM} if it corresponds to a correctly executed computation that begins with the correct initial configuration regardless of whether it accepts or not. We also omit the terms for the boundary conditions described for the finite chain and cycle. Instead we are going to add a new term $H_{\text{size},N}$ which will be energetically favorable to segments of length N.

Let $\mathcal{S}$ be the subspace spanned by all well-formed states in the standard basis which have segments and cannot evolve via forward or backwards transitions to a state which does have an illegal pair. Lemmas 5.4 and 5.5 still apply since we have not changed the transition rules. Thus, $\mathcal{S}$ is the set of states which can be obtained by starting each segment in the correct initial configuration (with an arbitrary quantum state on the witness tape) and applying any number of transition rules to each segment. Since the transition rules do not alter the segments, H_{TM} is closed over $\mathcal{S}$ as it was for the chain. The final Hamiltonian H will be the sum of H_{TM} and a set of terms which are all diagonal in the standard basis, which means that $\mathcal{S}$ will be closed under H as well. We will study $H|_{\mathcal{S}}$.

We will add in another Hamiltonian $H_{\text{size},N}$ that will be designed to be energetically favorable to segments for some specific size N. We first construct a Hamiltonian H of the form $H_{TM} + H_{\text{size},N}/p(N)$ for some polynomial in N. In the final Hamiltonian, there will also be a term which penalizes rejecting computations. $H_{\text{size},N}$ will have a fixed form (i. e., constant size to specify) except there will be several coefficients which are inverse polynomials in N and therefore require $O(\log N)$ bits to specify.

Since all two-particle terms are zero on the pair $><$, we can omit the two-particle terms which span two segments when considering $H|_{\mathcal{S}}$. Now H can be divided into a sum of terms, each of which acts on particles entirely within a segment. Let H^i be the sum of the terms which act on particles within segment i. Let l denote the length of the segment. We can define H^i_{size} and H^i_{TM} similarly. An eigenstate of H in $\mathcal{S}$ is then a tensor product of eigenstates of each H^i acting on the particles in segment i. The energy is the sum of the energies of each H^i on their corresponding eigenstate. The eigenstates for a segment of length l are exactly the same as the bracketed eigenstates for a chain of the same length. We know that if $l \neq 3 \bmod 4$, then the ground state has energy at least $\Omega(1/l^3)$.

We are now ready to define the final component of H. Define T_l to be the number of clock states for a finite chain or segment of length l. We determined in Section 5 that $T_l = 4(l-2)^2$. Note that the symbol $0\triangleright$ appears in exactly $l-2$ clock states.

$$H_{\text{size},N} = \frac{1}{N} I - 2|<\rangle\langle<| + \frac{T_N}{N-2}\left(|0\triangleright\rangle\langle 0\triangleright|\right). \tag{8.1}$$

We will analyze the ground state energy of a segment as a function of its length. We will need to use the Projection Lemma from [21] which will allow us to focus on the ground space of H^i_{TM}. Define $|\phi^l_Y\rangle$ to be the state that corresponds to the uniform superposition of all states that can be obtained by applying forward transition rules to the correct initial state with quantum state Y on the witness tape. The ground space of H^i_{TM} is within the span of all the $|\phi^l_Y\rangle$. Note that $\langle\phi^l_Y|H^i_{\text{size},N}|\phi^l_Y\rangle$ is the same for all Y since $H_{\text{size},N}$ depends only on the clock tracks.

Lemma 8.3. *[21] Let $H = H_1 + H_2$ be the sum of two Hamiltonians acting on a Hilbert space $\mathcal{H} = \mathcal{T} + \mathcal{T}^{\perp}$. The Hamiltonian H_2 is such that $\mathcal{T}$ is a zero eigenspace for H_2 and the eigenvectors in $\mathcal{T}^{\perp}$ have value at least $J > 2\|H_1\|$. Then*

$$\lambda(H_1|_{\mathcal{T}}) - \frac{\|H_1\|^2}{J - 2\|H_1\|} \leq \lambda(H) \leq \lambda(H_1|_{\mathcal{T}}),$$

where $\lambda(H)$ is the ground energy of H.

Corollary 1. There is a polynomial $p(N)$ such that $p(N)$ is $O(N^7)$ and for any segment of size $l \leq 2N$ and $H^i = H^i_{TM} + H^i_{\text{size},N}/p(N)$,

$$p(N)\lambda(H^i|_{\mathcal{S}}) \geq \langle \phi^l_Y | H^i_{\text{size},N} | \phi^l_Y \rangle - 1/2N^2 .$$

Proof. We use the projection lemma with $H_2 = p(N)H^i_{TM}$ and $H_1 = H^i_{\text{size},N}$. Note that H_1 need not be positive, although it does need to be positive on $\mathcal{T}$ in order to yield a non-trivial lower bound. We need to establish that $\|H^i_{\text{size}}\| = O(N)$. Since $l \leq 2N$, the first term is $O(1)$. The Hilbert space $\mathcal{S}$ is the set of all well-formed, bracketed states for that segment, so there can be at most one $\textcircled{0\rhd}$ site. Thus the second two terms in $H^i_{\text{size},N}$ are at most T_N/N for any state in $\mathcal{S}$, which is also $O(N)$. The spectral gap of H^i_{TM} is $\Omega(1/N^3)$, so we can choose $p(N)$ so that $p(N)$ is $O(N^7)$ and J (the spectral gap of $p(N)H^i_{TM}$) is at least $2N^2\|H_1\|^2 + 2\|H_1\|$. Using Lemma 8.3, we can lower bound $p(N)\lambda(H^i|_{\mathcal{S}})$ by $\langle \phi^l_Y | H^i_{\text{size},N} | \phi^l_Y \rangle - 1/2N^2$. □

Note that we are not able to use the projection lemma for very large l because the $\Omega(1/N^3)$ gap will not be large enough. In the lemma below, we determine the ground state energy of a segment as a function of its length. Large l (greater than $2N$, including infinite l) are dealt with separately with an argument that does not require the projection lemma.

Lemma 8.4. *The operator H^i acting on the l particles of segment i restricted to well-formed bracketed states will have ground state energy at most* 0 *and spectral gap $\Omega(1/N^4)$ per particle for $l = N$. The ground state energy is $\Omega(1/N^9)$ per particle for any other value of l.*

Proof. We have assumed throughout that $N = 3 \bmod 4$. For any l such that $l \neq 3 \bmod 4$, the ground energy is at least $\Omega(1)$ which is at least $\Omega(1/N)$ per particle. We will restrict our attention therefore to l such that $l = 3 \bmod 4$, so for $l \neq n$, we know that $|l - n| \geq 4$.

We consider four different cases based on the size of the segment l.

$\mathbf{l = N}$:

Consider a state $|\phi^N_Y\rangle$. Then $\langle \phi^N_Y | H^i_{TM} | \phi^N_Y \rangle = 0$. Recall that $|\phi^N_Y\rangle$ is a uniform superposition of states. There are T_N distinct configurations represented in the support of $|\phi^N_Y\rangle$. Exactly $N-2$ of these contain a $\textcircled{0\rhd}$ site. All of them contain one particle in state $\textcircled{<}$. Therefore

$$\langle \phi^N_Y | H^i_{\text{size},N} | \phi^N_Y \rangle = \frac{N}{N} - 2 + \frac{T_N}{(N-2)} \frac{(N-2)}{T_N} = 0. \tag{8.2}$$

Any state $|\psi\rangle$ that is orthogonal to the subspace spanned by the $|\phi^N_Y\rangle$ and is also in the subspace spanned by the well-formed states for segments of length N will have $\langle \psi | H_{TM} | \psi \rangle \geq \Omega(1/N^3)$ and $\langle \psi | H_{\text{size}}/p(N) | \psi \rangle \geq -1/N^7$. Thus, the spectral gap of H^i will be $\Omega(1/N^4)$ per particle.

$\mathbf{l > 2N}$:

Let ψ be a state in the standard basis that is well-formed, bracketed and has length l. We will only lower bound $\langle \psi | H^i_{\text{size}}/p(N) | \psi \rangle$. Since H^i_{TM} is non-negative, the lower bound will hold for all of H^i. Furthermore, we will omit the last term in H_{size} because this only adds to the energy. Every standard basis state in a bracketed well-formed segment of length l has exactly one occurrence of $\circledless$. Therefore the energy of a segment of length l will be at least $(l/N-2)/p(N)$. This is at least $(1/N-2/l)/p(N)$ per particle. Since $l \geq 2N+1$, this will be at least $\Omega(1/N^8)$ per particle. Note that this holds for infinite l as well.

2N ≥ l > n:

Since we can assume that $l = 3 \bmod 4$, we know that $l \geq N+4$. We will use the projection lemma for this case and show that $\langle \phi_Y^l | H^i_{\text{size}} | \phi_Y^l \rangle \geq 1/N^2$, which by Corollary 1 will be enough to lower bound $\lambda(H^i)$ by $1/(2N^2 p(N))$.

$$\begin{aligned} \langle \phi_Y^l | H^i_{\text{size}} | \phi_Y^l \rangle &= (l-N)\left(\frac{1}{N} - \frac{1}{l-2}\right) \\ &\geq (l-N)\left(\frac{1}{N} - \frac{1}{N+2}\right) \geq \frac{1}{N^2}. \end{aligned}$$

l < N:

Since we can assume that $l = 3 \bmod 4$, we know that $l \leq N-4$. Now we will use the projection lemma for this case and show that $\langle \phi_Y^l | H^i_{\text{size}} | \phi_Y^l \rangle \geq 1/N^2$, which will be enough to lower bound $\lambda(H)$ by $1/(2N^2 p(N))$.

$$\begin{aligned} \langle \phi_Y^l | H^i_{\text{size}} | \phi_Y^l \rangle &= (N-l)\left(\frac{1}{l-2} - \frac{1}{N}\right) \\ &\geq (N-l)\left(\frac{1}{N-2} - \frac{1}{N}\right) \geq \frac{4}{N(N-2)} \geq \frac{1}{N^2}. \end{aligned}$$

□

Finally, we add a term which adds energy 1 to any state which is not an accepting computation. If $N = 3 \bmod 4$ and there is a quantum witness that causes the verifier to accept with probability $(1-\varepsilon)$, then there is a state with energy at most ε/N^3 per particle. This is the state which corresponds to correct computations, using the good witness, within consecutive segments of length N. We can assume the acceptance probability has been amplified sufficiently to make ε small—smaller than $1/p(N)$ is sufficient. Note that the ground space has an N-fold degeneracy which corresponds to translations of this state.

Now suppose that there is no quantum witness which causes the verifier to accept on input $(N-3)/4$ with probability greater than ε. Consider a possibly infinite set of locations for the $\circledless$ and $\circledgtr$ particles and the subspace spanned by the standard basis states corresponding to these assignments. We will analyze the subspace spanned by these standard basis states and argue that the ground state energy of the Hamiltonian restricted to each such subspace is at least $\Omega(1/N^3)$ per particle. Therefore, we assume now that the locations of the particles in the $\circledless$ or $\circledgtr$ state are fixed. We can partition the infinite chain into maximal pieces whose leftmost particle is in state $\circledless$ and such that the piece has no other particles in state $\circledless$. Note that it may be that the leftmost piece extends infinitely to the left. We know that any piece that is longer than $2N$ incurs a cost of at least $\Omega(1/N^8)$ per particle from $H_{\text{size},N}$ alone. This follows from the same analysis for proper segments of length more than $2N$ given above. Now suppose that a piece has length at most $2N$. If it is not a proper segment (i. e., beginning with a $\circledless$, ending with a $\circledgtr$, and no

⧀ or ⧁ particles in between), then there must be at least one illegal pair. The illegal pair will cause an energy cost of 1. Since the energy from $H_{\text{size},N}/p(N)$ will be at least $-2/p(N)$, the total energy per particle will be $\Omega(1)$. We also know that if the piece is a proper segment and the length of the segment is some $l \neq N$ or $l = N$ and $N \neq 3 \bmod 4$, then there will be a cost of at least $\Omega(1/N^9)$ per particle on that segment. Finally, even if $l = N$ and $N = 3 \bmod 4$, every quantum witness will cause the verifier to reject on input $(N-3)/4$ with probability at least $(1-\varepsilon)$. When ε is small enough (polynomially small is sufficient), using standard QMA-completeness proof techniques (see, e. g. [2]), we can show that there will be an energy penalty of at least $\Omega(1/T_N^2)$ for that segment, which is $\Omega(1/N^5)$ per particle.

9 Conclusion

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

More generally, one can ask which Hamiltonians are computationally hard and which are easy. Our approach provides a framework to investigate the question, although our techniques are not powerful enough to answer it except in special cases. Perhaps hard Hamiltonians are pervasive for particles of large enough dimension, or perhaps they are always outlying exceptional points. The computational hardness structure of the space of Hamiltonians is clearly somewhat complicated. For instance, for the infinite chain discussed in Section 8, the hard class of Hamiltonians considered form a sequence (as $N \to \infty$) that converges to a Hamiltonian that is not itself hard: It lacks the H_{size} term, so one can have an infinite chain with no segments or control particles, and the ground state energy is 0.

It would also be nice to understand the finite-temperature behavior of these systems better. The hard constructions should have a very complicated landscape of energy states, and the actual ground states are highly sensitive to the exact number of particles. This suggests that even at finite temperature, these systems should have complicated dynamics and behave like a spin glass, but it is not completely clear.

There remain many additional variants of tiling and Hamiltonian problems that we have not studied. Probably for most variations (e. g., triangular lattice instead of square lattice), we can expect similar results. However, there are some borderline cases for which the answer is not clear. The case of WEIGHTED TILING with reflection symmetry and constant allowed cost remains open for the four-corners or open boundary conditions. When we have translational invariance but no additional symmetry, 2-DIM TIH is as hard as 1-DIM TIH, but the construction for 1-DIM TIH with reflection invariance does not generalize to 2-DIM TIH with rotational invariance. Can rotationally- and translationally-invariant Hamiltonians be hard in two or more dimensions? Also, the hardness result for ITIH uses quantum properties, and we do not have a similar classical result.

Another interesting avenue to pursue would be to apply a similar idea to other problems. For instance, the game of go produces a PSPACE-complete problem [29]. However, the computational problem GO is defined by asking whether black can force a win given a particular board configuration as input. However, there is no guarantee that these board configurations would appear in a regular game of go, which starts from a blank board or with a small number of stones already placed as a handicap. A more natural problem arising from GO is to ask who wins with optimal play when starting with a particular fixed handicapping rule. The handicapping rule should specify the number and locations of initial stones as a function of the board size. As with our tiling and Hamiltonian problems, the only thing that varies is the size; the rules and initial configuration are fixed. Thus, we would wish to show that this variant of GO is EXPSPACE-complete. Our techniques will not solve this problem, but at least our result points the way to ask the right question.

Acknowledgements

We would like to thank John Preskill for pointing out the connection to *N*-REPRESENTABILITY, Cris Moore and Igor Pak for some information about classical tiling problems, and Xiao-Gang Wen and Ignacio Cirac for suggesting some of the open problems mentioned in the conclusion. The comments of two anonymous referees were also very helpful in pointing us towards relevant literature on tiling, improving our proofs for one-dimensional tiling, and pointing out the fact that the one-dimensional tiling problems are in NC^1. Part of this work was done while both authors were visiting the Institute for Quantum Information at Caltech.

References

[1] DORIT AHARONOV, DANIEL GOTTESMAN, SANDY IRANI, AND JULIA KEMPE: The power of quantum systems on a line. In *Proc. 48th FOCS*, pp. 373–383. IEEE Comp. Soc. Press, 2007. [doi:10.1109/FOCS.2007.46] 32, 36, 83, 85

[2] DORIT AHARONOV, DANIEL GOTTESMAN, SANDY IRANI, AND JULIA KEMPE: The power of quantum systems on a line. *Comm. Math. Physics*, 287(1):41–65, 2009. Preliminary version at arXiv. Extended abstract in FOCS'07. [doi:10.1007/s00220-008-0710-3] 32, 95, 112

[3] SHAI BEN-DAVID, BENNY CHOR, ODED GOLDREICH, AND MICHAEL LUBY: On the theory of average case complexity. *J. Comput. System Sci.*, 44(2):193–219, 1992. Preliminary version in STOC'89. [doi:10.1016/0022-0000(92)90019-F] 32

[4] ROBERT BERGER: The undecidability of the domino problem. *Memoirs of the Am. Math. Soc.*, 66:1–72, 1966. 48

[5] ETHAN BERNSTEIN AND UMESH VAZIRANI: Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. Preliminary version in STOC'93. [doi:10.1137/S0097539796300921] 83, 96

[6] ALBERTO BERTONI, MASSIMILIANO GOLDWURM, AND VIOLETTA LONATI: The complexity of unary tiling recognizable picture languages: Nondeterministic and unambiguous cases. *Fundam. Inform.*, 91(2):231–249, 2009. Preliminary version in STACS'07. [doi:10.3233/FI-2009-0042] 33, 38

[7] JEAN-PHILIPPE BOUCHAUD AND MARC MÉZARD: Self induced quenched disorder: A model for the glass transition. *J. Phys. I France*, 4(8):1109–1114, 1994. Preprint at arXiv. [doi:10.1051/jp1:1994240] 37

[8] DAVID DEUTSCH: Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Royal Society of London Series A*, 400(1818):97–117, 1985. [doi:10.1098/rspa.1985.0070] 83

[9] PETER VAN EMDE BOAS: The convenience of tilings. In ANDREA SORBI, editor, *Complexity, Logic, and Recursion Theory*, pp. 331–363. Marcel Dekker Inc, 1997. 33

[10] MARTIN FÜRER: The computational complexity of the unconstrained limited domino problem (with implications for logical decision problems). In *Logic and Machines: Decision and Complexity*, volume 171 of *LNCS*, pp. 312–319. Springer, 1984. [doi:10.1007/3-540-13331-3_48] 33, 34

[11] HANA GALPERIN AND AVI WIGDERSON: Succinct representations of graphs. *Inf. Control*, 56(3):183–198, 1983. [doi:10.1016/S0019-9958(83)80004-7] 33

[12] DANIEL GOTTESMAN AND SANDY IRANI: The quantum and classical complexity of translationally invariant tiling and Hamiltonian problems, 2009. Conference version in FOCS'09. [arXiv:0905.2419] 31, 39

[13] ERICH GRÄDEL: Dominoes and the complexity of subclasses of logical theories. *Ann. Pure Appl. Logic*, 43(1):1–30, 1989. [doi:10.1016/0168-0072(89)90023-7] 33

[14] ALBERT E. INGHAM: On the difference between consecutive primes. *Quarterly Journal of Mathematics (Oxford Series)*, 8(1):255–266, 1937. [doi:10.1093/qmath/os-8.1.255] 52

[15] SANDY IRANI: Ground state entanglement in one dimensional translationally invariant quantum systems. *J. Math. Phys.*, 51(2):022101, 2010. Preprint at arXiv. [doi:10.1063/1.3254321] 36

[16] 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. *New J. Phys.*, 10(9):093004, 2008. Preprint at arXiv. [doi:10.1088/1367-2630/10/9/093004] 36

[17] EMMANUEL JEANDEL AND PASCAL VANIER: Periodicity in tilings. In *14th Internat. Conf. Developments in Language Theory (DLT'10)*, pp. 243–254. Springer, 2010. Preprint at arXiv. [doi:10.1007/978-3-642-14455-4_23] 39

[18] NEIL D. JONES AND ALAN L. SELMAN: Turing machines and the spectra of first-order formulas. *J. Symbolic Logic*, 39(1):139–150, 1974. Available at JSTOR. Preliminary version in STOC'72. 33

[19] A. S. KAHR, EDWARD F. MOORE, AND HAO WANG: Entscheidungsproblem reduced to the $\forall\exists\forall$ case. *Proc. Natl. Acad. Sci. USA*, 48(3):365–377, 1962. JSTOR. 44

[20] ALASTAIR KAY: Computational power of symmetric Hamiltonians. *Phys. Rev. A*, 78(1):012346, 2008. Preprint at arXiv. [doi:10.1103/PhysRevA.78.012346] 36

[21] JULIA KEMPE, ALEXEI KITAEV, AND ODED REGEV: The complexity of the local Hamiltonian problem. *SIAM J. Comput.*, 35(5):1070–1097, 2006. Preliminary version in FSTTCS'04. [doi:10.1137/S0097539704445226] 109

[22] ALEXEI KITAEV, ALEXANDER H. SHEN, AND MICHAEL N. VYALYI: *Classical and Quantum Computation*. AMS, 2002. [ACM:863284] 35, 83

[23] LEONID A. LEVIN: Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986. Preliminary version in STOC'84. [doi:10.1137/0215020] 32

[24] HARRY R. LEWIS: Complexity of solvable cases of the decision problem for the predicate calculus. In *Proc. 19th FOCS*, pp. 35–47. IEEE Comp. Soc. Press, 1978. [doi:10.1109/SFCS.1978.9] 33

[25] HARRY R. LEWIS AND CHRISTOS H. PAPADIMITRIOU: *Elements of the Theory of Computation*. Prentice Hall, 2 edition, 1997. [ACM:549820] 33

[26] YI-KAI LIU, MATTHIAS CHRISTANDL, AND FRANK VERSTRAETE: Quantum computational complexity of the *N*-representability problem: QMA complete. *Phys. Rev. Lett.*, 98(11):110503, 2007. Preprint at arXiv. [doi:10.1103/PhysRevLett.98.110503] 41

[27] DANIEL NAGAJ AND PAWEL WOCJAN: Hamiltonian quantum cellular automata in one dimension. *Phys. Rev. A*, 78(3):032311, 2008. Preprint at arXiv. [doi:10.1103/PhysRevA.78.032311] 36

[28] ROBERTO OLIVEIRA AND BARBARA M. TERHAL: The complexity of quantum spin systems on a two-dimensional square lattice. *Quantum Information & Computation*, 8(10):900–924, 2008. Preprint at arXiv. [ACM:2016987] 36

[29] CHRISTOS H. PAPADIMITRIOU: *Computational Complexity*. Addison-Wesley, 1995. 33, 39, 43, 47, 112

[30] CHRISTOS H. PAPADIMITRIOU AND MIHALIS YANNAKAKIS: A note on succinct representations of graphs. *Inf. Control*, 71(3):181–185, 1986. [doi:10.1016/S0019-9958(86)80009-2] 33

[31] ROHIT PARIKH: On context-free languages. *J. ACM*, 13(4):570–581, 1966. [doi:10.1145/321356.321364] 56

[32] PAUL W. K. ROTHEMUND AND ERIK WINFREE: The program-size complexity of self-assembled squares. In *Proc. 32nd STOC*, pp. 459–468. ACM Press, 2000. [doi:10.1145/335305.335358] 33

[33] LESLIE G. VALIANT: The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979. [doi:10.1137/0208032] 33

[34] HAO WANG: Proving theorems by pattern recognition II. *Bell Systems Technical Journal*, 40:1–41, 1961. 33

[35] ERIK WINFREE: *Algorithmic Self-Assembly of DNA*. Ph. D. thesis, California Institute of Technology, 1998. 33

AUTHORS

Daniel Gottesman
Faculty Member
Perimeter Institute, Waterloo, Canada
dgottesman@perimeterinstitute.ca
http://www.perimeterinstitute.ca/personal/dgottesman/

Sandy Irani
Professor
Computer Science Department
University of California, Irvine, USA
irani@ics.uci.edu
http://www.ics.uci.edu/~irani

ABOUT THE AUTHORS

DANIEL GOTTESMAN graduated from Caltech in 1997; his advisor was John Preskill. He is best known for his work on stabilizer codes and on teleportation of quantum gates. He has also worked on a variety of other quantum information topics, particularly fault-tolerant quantum computation, quantum cryptography, and quantum complexity.

SANDY IRANI graduated from UC Berkeley in 1991; her advisor was Richard Karp. Her research has focused on the application of algorithm design and analysis to computing systems. In particular, she has specialized in the area of on-line algorithms and their applications to scheduling and resource allocation. In the last few years she has been working in quantum computation.