Translationally Invariant Constraint Optimization Problems

Dorit Aharonov
Department of Computer Science and Engineering, Hebrew University, Jerusalem, Israel

Sandy Irani
Department of Computer Science, University of California Irvine, CA, USA
The Simons Institute for the Theory of Computing, University of California Berkeley, CA, USA

Abstract

We study the complexity of classical constraint satisfaction problems on a 2D grid. Specifically, we consider the computational complexity of function versions of such problems, with the additional restriction that the constraints are *translationally invariant*, namely, the variables are located at the vertices of a 2D grid and the constraint between every pair of adjacent variables is the same in each dimension. The only input to the problem is thus the size of the grid. This problem is equivalent to one of the most interesting problems in classical physics, namely, computing the lowest energy of a classical system of particles on the grid. We provide a tight characterization of the complexity of this problem, and show that it is complete for the class $\mathrm{FP}^{\mathrm{NEXP}}$. Gottesman and Irani (FOCS 2009) also studied classical constraint satisfaction problems using this strong notion of translational-invariance; they show that the problem of deciding whether the cost of the optimal assignment is below a given threshold is NEXP-complete. Our result is thus a strengthening of their result from the decision version to the function version of the problem. Our result can also be viewed as a generalization to the translationally invariant setting, of Krentel's famous result from 1988, showing that the function version of SAT is complete for the class $\mathrm{FP}^{\mathrm{NP}}$.

An essential ingredient in the proof is a study of the computational complexity of a gapped variant of the problem. We show that it is NEXP-hard to approximate the cost of the optimal assignment to within an additive error of $\Omega(N^{1/4})$, where the grid size is $N \times N$. To the best of our knowledge, no gapped result is known for CSPs on the grid, even in the non-translationally invariant case. This might be of independent interest. As a byproduct of our results, we also show that a decision version of the optimization problem which asks whether the cost of the optimal assignment is odd or even is also complete for $\mathrm{P}^{\mathrm{NEXP}}$.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Constraint satisfaction, Tiling, Translational-invariance

Digital Object Identifier 10.4230/LIPIcs.CCC.2023.23

Related Version *Full Version*: https://arxiv.org/abs/2209.08731

Acknowledgements We are grateful to the Simons Institute for the Theory of Computing, at whose program on the "The Quantum Wave in Computing" this collaboration began.

1 Introduction

More than half a century ago, Cook and Levin inaugurated the field of NP-completness. The fact that the Constraint Satisfaction Problem (CSP) is NP-complete has been the cornerstone of our understanding and approach to important optimization problems arising in countless applications. However, the importance of CSP and its NP-completeness stems not only from its central role in studying the complexity of optimization problems; in fact, the computational complexity of CSP is of major importance to physics as well.

In classical many body physics, the most basic notion is the local Hamiltonian, which expresses the total energy of a system of particles. It turns out that this local Hamiltonian can be viewed as a CSP. The energy of the system is written in such a Hamiltonian as the

sum of terms, each of which describes the energy interaction between constant-sized clusters of particles. These terms can be viewed as local constraints and finding whether the lowest energy state of such a system is below a given threshold or above it, is a special case of CSP. The classical local Hamiltonian problem was famously shown to be NP-complete in many cases by Barahona and others [6, 13], and the understanding of its complexity for a variety of Hamiltonians such as the Ising model, the Potts model, and more, has had a fundamental impact in several research areas in physics, including statistical mechanics and mathematical physics.

The theory of NP-completeness also has a natural generalization to the quantum setting. Like classical NP-completeness, the study of quantum NP-completeness has had a tremendous impact on our understanding of the relevant field in physics, namely condensed matter physics. More specifically, the Cook-Levin theorem was generalized by Kitaev [14] roughly 25 years ago to show that the following problem is QMA-complete: Given a local Hamiltonian with quantum energy interactions describing the energy in a quantum many-body system, decide whether the ground energy is above some value or below another. This problem had been intensely studied in recent years [14, 9, 16, 2]; importantly, over the past two decades, the study of local Hamiltonians and their computational complexity has led to the birth of a new field called "Hamiltonian complexity", which studies problems related to condensed matter physics, through the computational lens [9]. Both in classical statistical mechanics as well as in condensed matter and many-body quantum physics, the importance of the computational perspective has become one of the fundamental underpinnings of research today.

From the physics point of view, however, the general CSP setting commonly studied in computer science, in which each constraint is specified separately and independently based on the particular optimization problem of interest, seems quite contrived. By and large, physicists study local Hamiltonians, be them classical or quantum, in a *translational-invariant* (TI) setting. In this scenario, the particles are located at the vertices of a geometric lattice and all the terms acting on adjacent pairs of particles along a particular dimension are *the same*. Quantum and classical Hamiltonians are used to model the energy interactions of particles in a material. If the material is uniform, then it is natural that the energy interaction between particles would be the same throughout. Thus, in order for a theory of CSPs to be relevant in physics, it must consider this translational-invariance requirement, which introduces many new complexity challenges.

The model most relevant to physics is TI in a very strong sense: the dimension of the individual particles and the Hamiltonian term acting on each pair of adjacent particles in a lattice are fixed parameters of the problem. When considering finite systems, the only input is an integer N indicating the size of the system. This set-up corresponds to the fact that in physics, different Hamiltonians represent completely different physical systems. For example, studying the ground energy (or some other quantity) in the so-called AKLT model [1] is considered to be a completely different problem than studying the same quantity in, say, the Ising model[1].

Important progress on the computational complexity of translationally-invariant CSPs has been made in recent years. In particular, Gottesman and Irani [12] studied TI CSPs both in the classical and in the quantum setting, and showed hardness results in both cases. Since the size of the grid can be given by logarithmically many bits, and there is no other input, one encounters an exponential factor compared to the standard version of CSP. Thus

[1] Some of the recent work on the quantum TI local Hamiltonian problem [7] adopt a weaker notion in which the input also includes the Hamiltonian term that is applied to each pair of particles, allowing the Hamiltonian to be tuned to the size of the system. This model is mainly considered in quantum Hamiltonian complexity, but have not been a topic of study in physics.

the results in [12] show NEXP- and QMA_{EXP}-completeness for the classical and quantum variants of the problem, respectively. A tightly related line of work studies TI infinite systems [3, 21, 8] and considers computability and computational complexity in that domain, namely in the so-called thermodynamic limit. Although the focus in the current work is on finite systems, constructions for finite systems have played an important role for the results in the thermodynamic limit. In particular, all the results in [3, 21, 8] use a finite construction layered on top of a certain type of aperiodic tiling of the infinite grid.

However, all the results mentioned above have studied a *decision version* of the CSP problem. In contrast, in classical as well as in quantum physics, when considering the local Hamiltonian, the main problem of interest is the problem of finding the lowest possible energy for the Hamiltonian over all possible states – namely, the ground energy, which is one of the most important notions in physics. Thus, when considering CSPs with a physics motivation in mind, it seems that the *function version* of the CSP is a more relevant version of the problem than the decision version. In this setting, the question is not whether all constraints can be satisfied, but what is the *maximum number* of constraints that can be satisfied by any assignment, or, in a weighted variant, what is the cost of the optimal assignment which minimizes the weighted sum of violated constraints. We note that computing the cost of the optimal solution is in fact also the more natural version of CSPs in many combinatorial applications (to give just two examples, max-cut and max independent set).

What is known about the computational complexity of the function version of CSPs? In 1988, Krentel [15] proved that the function problem for constraint satisfaction is FP^{NP}-complete. Krentel's proof is significantly more involved technically than that of the Cook-Levin's theorem which characterizes the complexity of the decision variant of CSPs. In stark contrast to the theory of decision problems and NP-completeness, the function version of CSP seems to have received significantly less attention in the TCS literature.

In particular, to the best of our knowledge, the computational complexity of function CSPs in the TI setting, has remained open. In this paper we provide a tight characterization of its complexity, and show that the function version of TI CSP on a 2-dimensional grid is complete for FP^{NEXP}. This result thus strengthens Krentel's construction for general CSPs to apply even TI systems for two and higher dimensions. The result is also a generalization of Gottesman-Irani who prove hardness for 2D TI systems for the standard decision problem, where one only needs to determine if the ground energy is below a given threshold. One of the key technical challenges in our result is to effectively create large ($\Theta(N^{\epsilon})$) costs on an $N \times N$ grid using only two constant-sized terms which apply one in the horizontal and one in the vertical direction. Thus, as a stepping stone to the more complex result for the function version of TI 2D CSPs, we show a fault-tolerant result which we believe is of interest on its own, namely that it is NEXP-complete to even approximate the ground energy of 2D TI CSPs to within an additive $\Theta(N^{1/4})$.

2 Problem Definitions, Results and Main Challenges

It is most convenient to present our results using the language of the weighted tiling problem, where we focus here on the two dimensional case[2]. In this tiling problem, one is asked to tile an $N \times N$ 2D grid with a set of 1×1 tiles. The tiles come in different colors and only some pairs of colors can be placed next to each other in either the horizontal or vertical directions. More precisely, a set of tiling rules $\mathcal{T}$ is a triple (T, δ_H, δ_V), where T is a finite set of tile *types*

[2] Our version of tiling is equivalent to the more common Wang tiles [19].

$T = \{t_1, \ldots, t_d\}$, and δ_H and δ_V are functions from $T \times T$ to $\mathbb{Z}$. For $(t, t') \in T \times T$, $\delta_h(t, t')$ is the cost of putting a tile of type t immediately to the left of a tile of type t' and $\delta_v(t, t')$ is the cost of putting a tile of type t immediately above a tile of type t'. Let $\lambda_0(\mathcal{T}(N))$ be the minimum cost of tiling an $N \times N$ grid with tiling rules $\mathcal{T}$. The goal is to tile the grid with minimal total cost. Note that this problem is directly analogous to a classical Hamiltonian in 2D. We first define a function version of the problem.

▶ **Definition 1.** $\mathcal{T}$-FWT (Function Weighted Tiling)
Input: *An integer N specified with $\lfloor \log N + 1 \rfloor$ bits*
Output: $\lambda_0(\mathcal{T}(N))$

▶ **Theorem 2. (Main)** *There exists a set of tiling rules $\mathcal{T}$ such that $\mathcal{T}$-FWT is FP^{NEXP}-complete.*

We note that the fact that the function problem is complete for 2D immediately implies that it is complete for any grid of dimension at least 2 since the 2D construction can be embedded into a higher dimensional grid. The 1D CSP case is poly-time computable using dynamic programming.

The upper bound in Theorem 2 is easy: it can be achieved by binary search with access to an oracle for the decision problem. For the lower bound, one encounters a challenge. The reduction must encode in the tiling rules the computation of a polynomial time TM (TM) with access to a NEXP oracle. If an instance given to the oracle is a *yes* instance, the computation of the verifier can be encoded into the tiling rules. However *no* instances cannot be directly verified in this way. Krentel's proof that the function problem of weighted SAT is $\mathrm{FP}^{\mathrm{NP}}$-complete [15] overcomes this challenge; let us recall it and then explain the problem in carrying it over to the TI setting. Krentel uses an accounting scheme [15, 17] that applies a cost to every string z representing guesses for the sequence of responses to all the oracle queries made. The accounting scheme needs to ensure that the minimum cost z is equal to the correct sequence of oracle responses, $\tilde{z}$. *yes* and *no* guesses are treated differently, due to the fact that the verifier can check *yes* instances (and thus incorrect *yes* guesses can incur a very high cost), but *no* guesses, cannot be directly verified. In Krentel's scheme, *no* guesses incur a more modest cost, whether correct or not, and their cost must decrease exponentially. This is because the oracle queries are adaptive; an incorrect oracle response could potentially change all the oracle queries made in the future and so it is important that the penalty for an incorrect guess on the i^{th} query is higher than the cost that could potentially be saved on all future queries. The weights on clauses that implement this accounting scheme are multiplied by a large power of two to ensure that they are the dominant factor in determining the optimal assignment.

The difficulty in applying Krentel's accounting scheme in the TI setting is that the costs must grow with the size of the input. Therefore, it is not possible to apply the costs directly into the tiling rules which are of fixed constant size. A natural attempt to circumvent the problem is to assign the required large penalty by many tiles, each of which would acquire a constant penalty; however, the problem in implementing this approach is that Cook-Levin type reductions from computations to tilings are very brittle, as a single error can potentially derail the entire computation. For example, imagine inserting a row that does not have a TM head. There will be a single fault where the head disappears from one row to the next, but every row thereafter will contain the unchanging contents of the TM tape without a head to execute a next step. This imposes a challenge since when enforcing large costs by using many tiles, or constraints, we need to make sure that many of these constraints are indeed violated in order to incur the required large penalty.

We provide a construction which circumvents this issue by exhibiting some *fault tolerance* properties. We thus prove what can be viewed as a gapped version or a hardness of approximation result, which is then a natural stepping stone to implementing the more intricate function required in Krentel's accounting scheme. To this end we define an approximation version of weighted tiling:

▶ **Definition 3.** $(\mathcal{T}, f)$-GWT (Gapped Weighted Tiling)
Input: *An integer N specified with $\lfloor \log N + 1 \rfloor$ bits. Two integers a and b such that $b - a \geq f(N)$.*
Output: *Determine whether $\lambda_0(\mathcal{T}(N)) \leq a$ or $\lambda_0(\mathcal{T}(N)) \geq b$.*

▶ **Theorem 4.** *There exists a set of tiling rules $\mathcal{T}$ such that $(\mathcal{T}, f)$-GWT is NEXP-complete for a function $f(n) = \Omega(N^{1/4})$.*

This shows that it is NEXP-hard to even approximate the cost of the optimal tiling to within an additive error that is $\Omega(N^{1/4})$. This can be viewed as a gapped version of the results of [12]; the proof constructs a reduction mapping the computation into a tiling such that even in the presence of $O(N^{1/4})$ faults, the computation encoded by the tiling is able to proceed and produce approximately correct results.

Theorem 4 is of potential interest on its own. It might resemble a PCP type result, but the model we consider differs from the standard PCP setting in two ways: the first is that the underlying graph is a grid, rather than a graph with much higher connectivity, and the second is translational-invariance. It is not possible to obtain a hardness of approximation result with an additive error that is linear in N (as one has in the PCP theorem) on any finite dimensional lattice because such graphs do not have the necessary expansion properties. For example, in 2D, one could divide the grid into $b \times b$ squares for $b = \Theta(\sqrt{\log N})$ and solve each square optimally in polynomial time. The resulting solution would be within an additive $N/\sqrt{\log N}$ of the optimal solution. To the best of our knowledge, no gapped version was proven before for CSP problem set on a constant dimensional grid, even without the TI restriction.

Finally, our results provide tight characterizations of the complexity of the following decision problem;

▶ **Definition 5.** $\mathcal{T}$-PWT (Parity Weighted Tiling)
Input: *An integer N specified with $\lfloor \log N + 1 \rfloor$ bits*
Output: *Determine whether $\lambda_0(\mathcal{T}(N))$ is odd or even.*

The proof is very similar to the proof of Theorem 2. The result on Parity Weighted Tiling illustrates that decision problems related to CSP can be complete for an oracle class just like the function problem. The crucial difference between the threshold decision problem (is the cost of the optimal solution less than t?) which is NEXP-complete and the parity problem which is $\mathrm{P}^{\mathrm{NEXP}}$-complete is that the parity problem still seems to require determining the optimal cost. This seems to make the characterization of its complexity as challenging as for the function version of the problem.

Organization. We next proceed to an overview of the proofs. We start with the setup of tiling rules and layers in Section 3. Overviews of the proofs of the Theorems are given in subsection 4. We end with related work and open questions in Section 5. The complete proofs are given in the full version of the paper [4].

3 Tiling Rules and Layers

We assume that there is a special tile denoted by □ which must be placed around the perimeter of the grid to be tiled. Moreover, no □ tile can be placed in the interior of the grid. We will return later to enforcing this condition in the context of the different problems. The tiles on the interior will be composed of multiple layers where each layer has its own set of tile types. A tile type for an internal tile in the overall construction is described by a tile type for each of the layers.

For ease of exposition, we allow our tiling rules to also apply to local *squares* of four tiles. This can easily then be translated to two-local constraints on tiles, as in our definition of the tiling problem. This simple transition is described in more detail in the full version. For the remainder of the paper our tiling rules include constraints on local squares of four tiles, as well as pairs of horizontal tiles.

If the four tiles in a square are all interior tiles, then each possible pattern of four square tiles within a layer will be designated as legal or illegal. The overall cost of placing four interior tiles in a local square together will be function of whether the square for each layer is legal or illegal. For the Gapped Weighted Tiling, the cost will be just the number of layers for which the square pattern is illegal. For the Function Weighted Tiling and Weighted Tiling Parity, illegal squares at different layers will contribute different amounts to the cost.

In general, a no-cost tiling of each Layer represents a computational process where each row represents the state of a TM. The computation reverses direction from one layer to the next. The rows of a tiling of an $N \times N$ grid will be numbered r_0 through r_{N-1} from bottom to top. When referring to the rows in a particular layer, we will exclude the border rows and order the rows according to the computation direction. So the first row of Layer 1, which proceeds from bottom to top, is row r_1 and the last row of Layer 1 is r_{N-2}. Layer 2 proceeds from top to bottom, so the first row for Layer 2 is r_{N-2} and the last row is r_1.

For the most part, the rules governing the tiling apply to the tile types within each individual layer. The different layers only interact at the lower and upper border of the grid. This is how the output of one process (on Layer i) is translated into the input for the next process (on Layer $i+1$). For example, a square may be illegal if the two lower tiles are □ □, and the two upper tiles violate certain constraints between the Layer i and Layer $i+1$ types. Some of the layers will also have additional constraints on which tiles can be next to each other in the horizontal direction. Each type of violated constraint is given a name described below.

▶ **Definition 6** (Faults in a Tiling). *An occurrence of any of the illegal patterns described in the constructions is called a* fault. *A tiling with no faults, will correspond to a* fault-free *computation.*

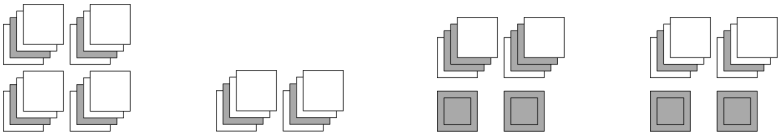
There will be some additional costs (described later) associated with a computation ending in a rejecting state. These are not considered faults because they can happen in correct computations. Figure 1 illustrates the different types of tiling constraints.

Illegal Computation Squares: For each layer, every pattern of four tile types will be designated as a *legal computation square* or an *illegal computation square*. In general, these rules enforce that the tiling within the layer represents a consistent execution of a TM. The full version of the paper [4] gives a set of rules to translate the rules of a TM into legal and illegal computation squares.

Illegal Pairs: Some of the layers will have additional constraints on which tiles can be placed next to each other in the horizontal direction. Each ordered pair of tiles types for that layer will be designated as a *legal pair* or an *illegal pair*.

Illegal Initialization Squares: For each layer, there are also some initialization rules that constrain the initial configuration of the TM. If the layer runs bottom to top, then these rules apply to r_0, which consists of all $\square$ tiles, and the first row of the layer. For example, if tile t_1 can not be immediately to the left of t_2 in the first row of Layer i, then the square with $\square$ $\square$ directly below t_1 t_2 is an *illegal initialization square* for Layer i. If the TM for the layer runs top to bottom, then the square with $\square$ $\square$ directly above t_1 t_2 in Layer i is illegal.

Illegal Translation Squares: Finally, we add rules that control how the last row of Layer i is translated to the first row of Layer $i+1$. If Layer i runs top to bottom, then the rules apply to rows r_0 and r_1. For example, if tile t in Layer i cannot be translated to t' in Layer $i+1$, then any square with a $\square$ directly below a tile whose Layer i type is t and whose Layer $i+1$ type is t' would be illegal. The translation rules can also apply to pairs of adjacent tiles. E.g., it could be illegal to have a square whose bottom two tiles are $\square$ $\square$ and whose top two tiles have t_1 t_2 in Layer i and t_3 t_4 in Layer $i+1$.



(a) Computation. **(b)** Illegal pairs. **(c)** Translation. **(d)** Initialization.

Figure 1 Interior tiles have four layers. Border tiles have one layer and are labeled with the $\square$ symbol. (a) An illegal computation square for Layer 2. The constraint applies to the four tile types for Layer 2 shown in gray. (b) An illegal pair for Layer 2. The constraint applies to the two adjacent tile types for Layer 2 shown in gray. (c) An illegal translation square from Layer 2 to Layer 3. The constraint applies to two border tiles and the tile types for Layers 2 and 3 for the other two interior tiles. (d) An illegal initialization square for Layer 2. The constraint applies to two border tiles and the Layer 2 tile types for the other two interior tiles.

4 Overview of Proofs

Thoeorem 4: Gapped Weighted Tiling. Recall that the standard encoding of a TM into tiling rules is very brittle in that a single fault can derail the entire computation. The most straight forward way to overcome this is using a construction which embeds many repetitions of the computation, so that many faults would be required to derail a large number of those computations. Multiple computations thus need to be set up and initiated, using a single faulty TM with TI rules. In our construction, this is achieved by a first stage of the computation (implemented in Layer 1, as we describe below), which, roughly, creates intervals in the top row of Layer 1, such that the independent repetitions of the computations will occur in different strips on the grid; the boundaries of the strips are determined by those intervals. The difficulty is how to implement the initial set up using a single TM, in a fault tolerant way. We now describe the details.

The tiling rules for the first two layers, as well as the reduction mapping x to N are independent of the language $L \in$ NEXP, the language we are reducing from. Let V denote the exponential time verifier for L. In general tiles will be either *tape* tiles which encode a single symbol from the TM's tape alphabet or *head* tiles which encode both the state of the TM as well as the current tape symbol to which the head is pointing.

The TM computation represented in Layer 1 starts with two non-blank symbols and proceeds to write a sequence of intervals on the tape, where an interval is a sequence of B symbols bracketed on either side by a *delimiter* tile from the set $\{X, \overline{X}, \triangleleft, \triangleright\}$. The TM just repeatedly executes a single loop which we refer to as the *Outer Loop*. In one iteration of the Outer Loop, an additional B symbol is inserted into every interval and a new interval with no B's in the middle is added to the right end of the non-blank symbols. In a fault-free execution of the TM, after m iterations of the loop, there are $m+1$ intervals. The number of symbols in each interval (including the delimiter tiles on either end) is $m+2, m+1, m, \ldots, 2$. For $m = 4$, the row should look like:

□	$(q/\triangleleft)$	B	B	B	X	B	B	X	B	X	$\triangleright$	#	#	$\cdots$	#	□

When the top row of Layer 1 is translated to Layer 2, the head tile for the Layer 1 TM is translated to a tape tile (so the state information is lost) and a head tile is inserted on the left end of every interval. For example, an interval $X\ B\ B\ B\ \cdots\ B\ X$ at the end of Layer 1 is translated to $X\ (q_s/S)\ B\ B\ \cdots\ B\ T\ X$ in the first row of Layer 2. In Layers 2 and 3 the sizes and locations of the intervals do not change within a row unless the interval contains an illegal square. Thus, a single interval over all the rows of Layer 2 forms a vertical strip of tiles, and a separate, independent computation takes place within each strip. See Figure 2 for an example. Once the intervals are created on Layer 1, each computation on Layers 2 and 3 is fault-free unless the strip contains an illegal square. Thus, the number of illegal squares is at least the number of strips that fail to complete their computation correctly.

In Layer 2, the computation is just a binary counter TM that continually increments a binary counter. All the strips that do not contain an illegal square will have the same string x represented in the final row of Layer 2. The string x then serves as the input to the computation in Layer 3. The binary counter TM in Layer 2 runs for exactly $N-3$ steps. The reduction is the function that maps x to N, where the string x is written on the tape of a binary counting TM after $N-3$ steps. The full version [4] gives an exact formula mapping x to N and shows that the value of the number represented by the string x is $\Theta(N)$, the dimension of the grid. The idea of using a binary counting TM to translate the size of the grid to a binary input for a computation was used previously in [12]. Although since the construction in [12] had a gap of 1, only a single execution of the verifier was needed. Since we are trying to produce a gap of $f(N)$, we need at least $f(N)$ separate computations each of which simulates the verifier on input x.

Each interval $X\ (q_s/S)\ x\ B\ \cdots\ B\ T\ X$ is translated unchanged to Layer 3. The computation in each strip in Layer 3 simulates the verifier on input x using a witness that is guessed in the tiling. There is a final cost for any rejecting computation. If $x \in L$, it will be possible to tile each strip at 0 cost. If $x \notin L$, every strip will contain an illegal square or will incur a cost for the correct rejecting computation. Thus, the gap is essentially created by these parallel computations, each of which contributes a constant cost if $x \notin L$.

Since the sizes of the intervals go down to 0, some of the intervals will be too narrow to complete the computation in either Layers 2 or 3. If the head ever hits the right end of its interval, it transitions to an infinite loop, causing no additional cost. A standard padding argument (provided in detail in the full version [4]) guarantees that an interval need only be $\Theta(N^{1/4})$ wide to complete the computations in Layers 2 and 3. The analysis of Layer 1 then needs to guarantee that despite the faults, there will be sufficiently many sufficiently wide intervals.

The main challenge in the proof is in making the computation in Layer 1 fault-tolerant, meaning that each illegal pair or square cannot derail the computation too much. The horizontal rules in Layer 1 are critical for enforcing that this cannot happen. We show that a row in the tiling that has no illegal pairs corresponds to a sensible configuration of the TM.

In particular such a row has exactly one head tile that lies in between the $\triangleleft$ and $\triangleright$ tiles. Note that faults can still alter the computation in potentially strange ways. Nonetheless, we also show that starting from a row with no illegal pairs, the Layer 1 TM will be able to make progress, and after a sequence of fault-free steps (corresponding to a sequence of rows containing no illegal squares), the computation will perform a complete iteration of the loop. Since the number of illegal pairs and squares is bounded by $O(N^{1/4})$, there are enough complete iterations of the loop to ensure that the last row of Layer 1 has enough intervals that are wide enough to complete the computations in Layers $2, 3$.

By far the most technically involved part of the paper is the analysis of Layer 1. All of the results make use of a tight characterization of the difference between the final row in Layer 1 of a fault-free tiling and the final row of a tiling with faults. In fact, the result on Gapped Weighted Tiling could be established with looser bounds, but we provide the analysis once in a form that can be used for all the results in the paper. Section 4 describes more fully how this tight characterization is accomplished.

Theorem 2: Weighted Tiling Function. The hardness reduction for Function Weighted Tiling reduces from an oracle class. The function f is computed by a polynomial time TM M with access to an oracle for language $L' \in$ NEXP. Let V denote the exponential time verifier for L'. Using a standard padding argument (see for example Lemma 2.30 from [3]) we can assume that for a constant c of our choice, for every $|x| = n$, there is a $\overline{n} \leq cn$, such that the size of $f(x)$ is at most $\overline{n}$, and M makes at most $\overline{n}$ oracle calls to L'. Let z denote an $\overline{n}$-bit string denoting the responses to the oracle queries made on input x. With x and z fixed, the set of inputs to the oracle $(o_1, \ldots, o_{\overline{n}})$ is also determined. $V(o_j)$ is an indicator function denoting whether o_j is in L'. Note that since L' is in NEXP, if $V(o_j) = 1$, there exists a witness that will cause the verifier to accept and if $V(o_j) = 0$, V will always reject regardless of the witness. Define:

$$\mathcal{C}(x, z) = \sum_{j=1}^{\overline{n}} \left[(1 - z_j) \cdot 2^{\overline{n}-j} + z_j \cdot (1 - V(o_j)) \cdot 2^{\overline{n}}\right] \tag{1}$$

Let $f(x, z)$ be the output of TM M on input x with oracle responses z. Note that since $|f(x, z)| \leq \overline{n}$, $f(x, z) \leq 2^{\overline{n}}$. The construction will ensure that the minimum cost tiling for a particular x and z will be $2^{\overline{n}+5}\mathcal{C}(x, z) + 2^3 \cdot f(x, z)$. Note that $\mathcal{C}(x, z)$ represented in the $\overline{n}$ high-order bits of the cost has the necessary structure where the costs for a *no* oracle response decrease exponentially in j, the index of the oracle query. The cost for a *yes* guess will be 0 if the input to the oracle o_j is in fact in L' (i.e., $V(o_j) = 1$) and will be a very large cost of $2^{2\overline{n}+5}$ if o_j is not in L'. This function will guarantee that the overall cost is minimized when z is the correct string of oracle responses. In addition, the low order bits encode the output of the function $f(x, z)$. So if the minimum cost tiling can be computed, this will correspond to $f(x)$, which is $f(x, \bar{z})$, where $\bar{z}$ is the string of correct oracle responses. The factor of 8 ensures that even if the minimum cost is off by ± 3, the value of $f(x)$ can still be recovered.

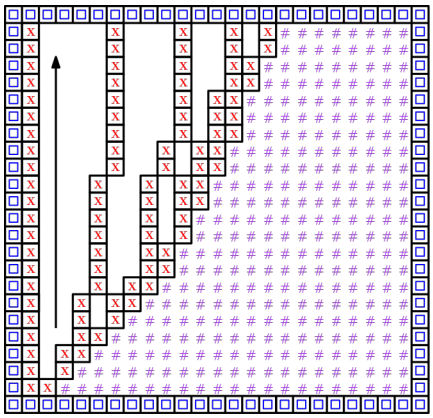
So far what we have described just implementing the original accounting scheme devised by Krentel. The challenge is to implement this cost function in 2D with TI terms. Note that since the tiling rules are fixed parameters of the problem, it is not possible to encode the cost function directly into the penalty terms. As with the Gapped Weighted Tiling problem the function is collectively computed by a set of parallel processes within each strip created by the intervals from Layer 1. However, instead of a threshold function which is either $+f(N)$ or 0, the parallel processes must collectively compute the more intricate function described above, which requires that the individual processes have some additional information.

We will describe first what happens in a fault-free computation (with no illegal pairs or computation squares) and then describe how fault-tolerance is enforced and proven. A schematic view of the construction ins given in Figure 2. The construction for Layers 1 and 2 are exactly the same as for the Gapped Weighted Tiling problem. Layer 1 creates a set of intervals. We definite the function $\mu(N)$ to denote the number of intervals on the tape if the TM for Layer 1 executes $N-3$ steps. If after $N-3$ steps, the computation just happens to finish at the end of an execution of the Outer Loop then the intervals have sizes (from left to right) $\mu(N)+1, \mu(N), \ldots, 2$. If the computation finishes in the middle of an execution of the Outer Loop, the actual sequence of interval sizes will be close to $\mu(N)+1, \mu(N), \ldots, 2$. The largest interval could have size $\mu(N)+2$ and there may be a couple missing values in the range where the current interval is being increased. A complete description of the possible deviations is given in the full version [4]. $\mu(N)$ is $\Theta(N^{1/4})$ and we show using a standard padding argument that for the constant c of our choice, all of the computations require at most $c\mu(N)$ space. This allows us to establish that at least half of the intervals will be large enough to complete the required computations.

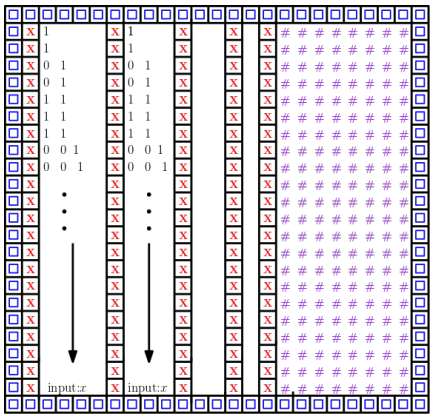
As in the previous construction, Layer 2 then executes a binary counting TM which results in the string x written to the left of each interval which is large enough to complete the computation. Note that Layer 1 is a *global* TM which executes a single process across the entire grid, while Layer 2 represents *local* computations within each strip. When x is translated from Layer 2 to Layer 3 it is augmented with a guess string z for the oracle queries. x. However, there is no guarantee that the guess for each interval is the same. Note that z can be arbitrary but it must be consistently the same for each interval. Layer 3 then executes a global TM which imposes a high penalty if the z strings in each strip are not all the same. This penalty is higher than the cost function for any z, so the lowest cost tiling will correspond to a configuration in which each strip has the same x and z.

Finally, in Layer 4, there is a local computation in each interval, each of which makes a $+1$ or 0 contribution towards the overall cost. The computation within each interval requires a unique tag in order to determine which term of the cost it will contribute to. The tag comes from the size of the interval. The computation begins with counting the number of locations in the interval. This can be accomplished by having the head shuttle back and forth between the two ends of the interval implementing both a unary and binary counter until the unary counter extends across the entire interval. The head returns to the left end of the interval and begins the next phase of the computation. Since the size of an interval is at most $O(N^{1/4})$ this phase of the computation will take at most $O(N^{1/2})$ steps.

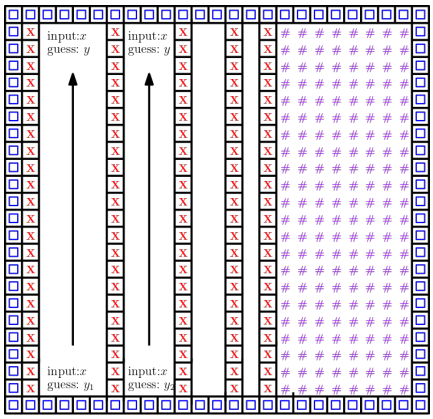
Now each computation has the same pair (x, z) and a its own integer r indicating the size of the interval. From x, the values of N and $\mu(N)$ can be determined. In a fault-free computation, the sizes of the intervals will decrease from left to right. Moreover, all interval sizes are in the set $\{\mu(N)+2, \mu(N)+1, \ldots, 2\}$ with at most one missing value from that set and at most two duplicates. Thus, the value $\mu(N)-r+2$, will be an almost unique identifier for each interval, starting with 0 or 1 on the left and increasing to the right. Using this tag, each interval determines which portion of the cost it will contribute to. The number of intervals assigned to compute a particular term in the cost will depend on the value of the term since each interval can contribute at most 1 to the overall cost. If an interval is assigned to check a *yes* guess ($z_k = 1$) the computation uses x and z to determine the k^{th} input to the oracle o_k, guesses a witness and simulates V on input o_k with the guessed witness. There is a cost of $+1$ if V rejects and 0 if V accepts. If o_k is in fact in L', there is a witness which will allow for a zero cost tiling withing that interval. If $o_k \notin L$, then every witness will lead to a $+1$ cost. Thus, the optimal set of witnesses will result in the minimum



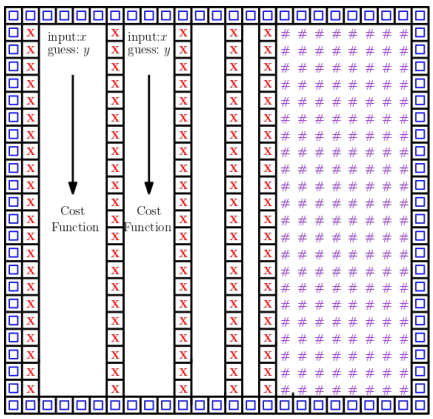
(a) In Layer 1 a single fault-tolerant TM creates the intervals that mark off the width of each strip where independent computations will take place in subsequent layers. The top row of Layer 1 is mapped onto the top row of Layer 2. The computation proceeds upwards from bottom to the top.



(b) In Layer 2, an independent computation takes place in each strip. The computation proceeds from top to bottom. Each strip executes a binary counter TM, so the height of the square is translated into a string x, which serve as the input to the computational problem.



(c) The bottom row of Layer 2 containing the string x is translated to Layer 3. In addition the tiling contains a guess y for the responses to the oracle query. Layer 3 then executes a global TM that proceeds from bottom to top. The computation results in a high cost if the guess strings y in each strip are not all the same.



(d) The independent computations in each strip collectively incur a total cost of $2^{\overline{n}+5}\mathcal{C}(x,z) + 2^3 \cdot f(x,z)$, where $\mathcal{C}(x,z)$ is denoted in Equation (1).

Figure 2 Schematic image showing the four layers in the construcion.

value for $2^{\overline{n}+5}\mathcal{C}(x,z)$. In addition, exactly $2^3 f(x,z)$ of the intervals will just transition to the rejecting state, incurring a cost of $+1$. The total cost due to those intervals is $2^3 f(x,z)$. For the remaining intervals, no cost is incurred.

The cost of a computation fault (illegal pair or square) is a constant that is larger than the cost of ending in a rejecting computation. Therefore, for each independent computation (in Layers 2 and 4) the optimal tiling will correspond to a correct computation which may or may not incur a cost for ending in a rejecting state. Technically, the most challenging part of the proof is to show that the process on Layer 1 which creates the intervals is fault-tolerant. The proof for Function Weighted Tiling requires stronger conditions than for Gapped Weighted

Tiling since we not only have to show that there are a large number of large intervals at the end of Layer 1 but we need to establish that the sequence of interval sizes is close to what one would have in a fault-free computation. To this end, we use a potential function A which captures how much a sequence of interval sizes $(s_1, s_2, \ldots, s_m)$ deviates from the expected sequence $(m+1, m, m-1, \ldots, 2)$. The main part of the proof is to show that each illegal square or pair can cause the value of A to increase by at most a constant amount. At the end of Layer 1, the ideal sequence of interval sizes is $(\mu(N)+1, \mu(N), \ldots, 2)$. Every interval size that is missing from the actual sequence of interval sizes has caused A to increase by at least a fixed amount which in turn corresponds to faults incurred in the computation. Thus, we show that it is more cost-effective to complete the computation correctly (and not incur the higher cost of a fault) and incur the smaller potential cost of a rejecting computation.

The most important measure of progress of the tiling/computation in Layer 1 is the number of times the encoded TM completes an iteration of the Outer Loop in which the size of every interval increases by 1 and a new interval of size 2 is added. Faults can potentially cause an iteration of the Outer Loop to take longer as they may force the head to shuttle back and forth more times which in turn could result in fewer iterations. Even in a fault-free computation, the number of steps per iteration increases with each iteration because there are more intervals. The analysis in the full version provides a lower bound on the number of times the loop is completed in relation to the number of completed loops in a fault-free computation. The proof is a delicate inductive argument which uses the fact that the increase in the running time of a loop is not accelerated too much with each additional fault.

Proof Overview for Parity Weighted Tiling. The proof for parity weighted tiling is very similar to the function problem. Suppose that a language $L \in \mathrm{P}^{\mathrm{NEXP}}$ is computed by a TM M with access to an oracle for $L' \in \mathrm{NEXP}$. Let $M(x, z)$ be the indicator function that is 0 if $M(x, z)$ accepts and 1 if $M(x, z)$ rejects. The overall cost computed by the collective computations is: $4\mathcal{C}(x, z) + M(x, z)$. The left-most interval computes $M(x, z)$ and results in a $+1$ cost in the case that M rejects. The remaining intervals which collectively compute Krentel's cost function all impose costs of $+2$ or 0. Thus the expression $4\mathcal{C}(x, z) + M(x, z)$ will guarantee that the minimum $\mathcal{C}(c, z)$ corresponds to the correct guess $\bar{z}$. Furthermore, the rightmost bit will be $M(x, z)$ which will cause the minimum cost to be odd or even, depending on whether M accepts.

5 Discussion, Related Work, and Open Problems

Despite the fact that the function version of classical local-Hamiltonians describes the task of the computational (classical) physicist much more naturally than decision problems, complexity of function problems was hardly studied even in the non-TI setting, in the literature of classical theory of computer science.

Recently, related results were discovered in the domain of quantum computational complexity. In particular, in [3], Aharonov and Irani use a construction for the function version of (finite) quantum local Hamiltonian as a component for a hardness result for the infinite 2D grid. More specifically, they prove that the problem of estimating the ground energy of a local Hamiltonian on a finite 2D grid, is hard for $\mathrm{FP}^{\mathrm{NP}}$. Importantly, their results do not imply the hardness result presented in this paper, and it seems impossible to extend their proof to deduce the classical hardness result of Theorem 2. Like [3] we implement Krentel's cost function using a fixed Hamiltonian term, but since their construction is quantum (as opposed to the classical construction in this paper), they are able to prove the

result using a completely different set of tools which do not carry over to the classical case. In quantum constructions, the lowest energy is an eigenvalue of a general Hermitian matrix and the matrix can be constructed to fine tune the ground energy to an inverse polynomial precision. In classical constructions, the total energy will be a sum over terms where each term is chosen from a constant-sized set of values determined by the finite horizontal and vertical tiling rules. This allows far less control in the classical setting over the precision of the minimum cost tiling.

Incidentally, note that the results for the quantum case proven in [3] are not tight, which follows from the fact that they use a quantum construction to obtain hardness for $\mathrm{FP}^{\mathrm{NEXP}}$, a classical complexity class. It seems challenging to make the characterization tight in the quantum case. In contrast to the class NP, the class QMA is a class of promise-problems and in simulating a $\mathrm{P}^{\mathrm{QMA}}$ machine, there is no guarantee that the queries sent to the QMA oracle will be valid queries. The cost/energy applied for a particular query will depend on the probability that a QMA verifier accepts on the provided input. If the input is invalid, then the probability of acceptance can be arbitrary. Thus, Krentel's cost function will potentially be an uncontrolled quantity. Typically in a reduction where we want to embed the output of a function into the value of the minimum energy, the low order bits of the energy are used to encode the output of the function. It's not clear how to do this without being able to control the binary representation of the minimum energy. Note that by embedding a classical computation in the Hamiltonian, the issue of invalid queries is circumvented.

Both [3] and [21] study the complexity of computing the ground energy density of infinite TI Hamiltonians to within a desired precision making use of the technique introduced by Cubitt, Prerez-Garcia, and Wolf which embeds *finite* Hamiltonian constructions of exponentially increasing sizes, into the 2D infinite lattice, using Robinson tiles. Robinson tiling rules [18] force an aperiodic structure on the tiling of the infinite plane, with squares of exponentially increasing size. The quantum construction of [3] layers a TI 1D Hamiltonian on top of one of the sides of all the squares. The classical construction of [21] layers a classical finite construction on each square. Neither work obtains tight results due to the same issue with invalid queries, although the two papers compromise in completely different ways. The primary technical innovation introduced in [21] is to devise a more robust version of Robinson tiles which ensures that the lowest energy state corresponds to a correct Robinson tiling, even though the cost of the classical finite construction layered on top may introduce a penalty. If it were possible to obtain an even more robust version of Robinson tiles, one potentially could layer the finite construction from the current paper on top the more robust constructions in the hopes of showing that computing the ground energy density of a classical TI Hamiltonian in the thermodynamic limit is complete for $\mathrm{EXP}^{\mathrm{NEXP}}$ under Karp reductions.

The results in this paper are also related to the work of Ambainis [5] which characterizes the complexity of measuring local observables of ground states of local Hamiltonians (APX-SIM), showing that the problem is complete for $P^{\mathrm{QMA}[\log n]}$. $P^{\mathrm{QMA}[\log n]}$ contains those problems that can be solved by a polynomial time classical TM with access to $O(\log n)$ queries to a QMA oracle. This type of question (determining a property of the ground state) is similar to our classical result about determining whether the cost of the optimal tiling is odd or even. The results on APX-SIM [5, 11, 10] are not hindered by the issue of invalid queries because the quantity being measured is not the actual energy itself. Note that the important point here is the property that distinguishes the state to be measured (minimum energy) is different than the local observable applied to the measured state. By contrast, computing the energy of the lowest energy state appears to be more difficult. The issue of invalid queries appears to be an obstacle, even when the Hamiltonian terms are position-dependent as in the constructions of [11, 10], as well as in the TI constructions in [3, 20].

Finally, it was mentioned earlier that the approximation problem considered here differs from the standard PCP setting in that the underlying graph is a grid and the terms are TI. It remains an open question as to whether there is a family of TI instances of constraint satisfaction on general graphs for which it is hard to estimate the optimal solution to within an additive $\Theta(N)$.

References

1 Ian Affleck, Tom Kennedy, Elliott H. Lieb, and Hal Tasaki. Rigorous results on valence-bond ground states in antiferromagnets. *Phys. Rev. Lett.*, 59:799–802, August 1987. doi:10.1103/PhysRevLett.59.799.

2 Dorit Aharonov, Daniel Gottesman, Sandy Irani, and Julia Kempe. The power of quantum systems on a line. *Communications in Mathematical Physics*, 287(1):41–65, January 2009. doi:10.1007/s00220-008-0710-3.

3 Dorit Aharonov and Sandy Irani. Hamiltonian complexity in the thermodynamic limit. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 750–763. ACM, 2022. doi:10.1145/3519935.3520067.

4 Dorit Aharonov and Sandy Irani. Translationally invariant constraint optimization problems, 2022. arXiv:2209.08731.

5 Andris Ambainis. On physical problems that are slightly more difficult than qma. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 32–43, 2014. doi:10.1109/CCC.2014.12.

6 F Baharona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, 1982.

7 Johannes Bausch, Toby Cubitt, and Maris Ozols. The complexity of translationally invariant spin chains with low local dimension. *Annales Henri Poincaré*, 18(11):3449–3513, October 2017. doi:10.1007/s00023-017-0609-7.

8 Toby S. Cubitt, David Perez-Garcia, and Michael M. Wolf. Undecidability of the spectral gap. *Nature*, 528(7581):207–211, December 2015. doi:10.1038/nature16059.

9 Sevag Gharibian, Yichen Huang, Zeph Landau, and Seung Woo Shin. Quantum hamiltonian complexity. *Foundations and Trends® in Theoretical Computer Science*, 10(3):159–282, 2015. doi:10.1561/0400000066.

10 Sevag Gharibian, Stephen Piddock, and Justin Yirka. Oracle complexity classes and local measurements on physical hamiltonians. *arXiv*, 2019. arXiv:1909.05981.

11 Sevag Gharibian and Justin Yirka. The complexity of simulating local measurements on quantum systems. *Quantum*, 3:189, September 2019. doi:10.22331/q-2019-09-30-189.

12 Daniel Gottesman and Sandy Irani. The quantum and classical complexity of translationally invariant tiling and hamiltonian problems. *Theory of Computing*, 9(2):31–116, 2013. doi:10.4086/toc.2013.v009a002.

13 Sorin Istrail. Statistical mechanics, three-dimensionality and np-completeness. i. universality of intractability for the partition function of the ising model across non-planar lattices. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 87–96, January 2000. doi:10.1145/335305.335316.

14 A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, USA, 2002.

15 Mark W. Krentel. The complexity of optimization problems. In Alan L. Selman, editor, *Structure in Complexity Theory*, pages 218–218, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

16 R. Oliveira and B. Terhal. The complexity of quantum spin systems on a two-dimensional square lattice. *arXiv*, 2005. arXiv:quant-ph/0504050.

17 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

18 Raphael Robinson. Undecidability and nonperiodicity for the tilings of the plane. *Invent. Math.*, 12:177–209, 1971.

19 Hao Wang. Proving theorems by pattern recognition. *Communications of the ACM*, 3(4):220–234, 1960.

20 James D. Watson, Johannes Bausch, and Sevag Gharibian. The complexity of translationally invariant problems beyond ground state energies. *arXiv*, 2020. `arXiv:2012.12717`.

21 James D. Watson and Toby S. Cubitt. Computational complexity of the ground state energy density problem. *arXiv*, 2021. `arXiv:2107.05060`.