On the Representation of Boolean and Real Functions as Hamiltonians for Quantum Computing

Stuart Hadfield*

Quantum Artificial Intelligence Lab, NASA Ames Research Center, Moffett Field, CA 94035

USRA Research Institute for Advanced Computer Science, Mountain View, CA 94043

Department of Computer Science, Columbia University, New York, NY 10027

December 30, 2021

Abstract

Mapping functions on bits to Hamiltonians acting on qubits has many applications in quantum computing. In particular, Hamiltonians representing Boolean functions are required for applications of quantum annealing or the quantum approximate optimization algorithm to combinatorial optimization problems. We show how such functions are naturally represented by Hamiltonians given as sums of Pauli Z operators (Ising spin operators) with the terms of the sum corresponding to the function's Fourier expansion. For many classes of Boolean functions which are given by a compact description, such as a Boolean formula in conjunctive normal form that gives an instance of the satisfiability problem, it is #P-hard to compute its Hamiltonian representation, i.e., as hard as computing its number of satisfying assignments. On the other hand, no such difficulty exists generally for constructing Hamiltonians representing a real function such as a sum of local Boolean clauses each acting on a fixed number of bits as is common in constraint satisfaction problems. We show composition rules for explicitly constructing Hamiltonians representing a wide variety of Boolean and real functions by combining Hamiltonians representing simpler clauses as building blocks, which are particularly suitable for direct implementation as classical software. We further apply our results to the construction of controlled-unitary operators, and to the special case of operators that compute function values in an ancilla qubit register. Finally, we outline several additional applications and extensions of our results to quantum algorithms for optimization. A goal of this work is to provide a *design toolkit for quantum optimization* which may be utilized by experts and practitioners alike in the construction and analysis of new quantum algorithms, and at the same time to provide a unified framework for the various constructions appearing in the literature.

1 Introduction

A basic requirement of many quantum algorithms is the ability to translate between mathematical functions acting on a domain, typically strings of bits, and quantum mechanical Hamiltonian operators acting on qubits. In particular, mapping Boolean or real functions to Hamiltonians has important applications in quantum algorithms and heuristics for solving decision or optimization problems such as quantum annealing and adiabatic quantum optimization [1–3], or the quantum approximate optimization algorithm and quantum alternating operator ansatz (QAOA) [4–6], or the related variational quantum eigensolver [7]. Explicit Hamiltonian constructions for the application of these algorithms to a variety of prototypical problems can be found in [6, 8], though prior work has mostly focused on reductions to quadratic Hamiltonians at the expense of additional qubits, for example in the penalty term approach of quantum annealing; we explain

*email: stuart.hadfield@nasa.gov / shadfield@usra.edu

how direct (not necessarily quadratic) mappings are desirable for quantum gate model algorithms. Such quantum algorithms are promising, in particular, as possible paths towards performing useful computation on near-term quantum computing devices. Indeed, decision and optimization problems are ubiquitous across science and engineering, yet often appear to be computationally difficult. Despite years of investigation, efficient algorithms often remain elusive [9, 10]. Hence, the potential for new approaches to tackling these problems on quantum computers is an exciting development.

Nevertheless, the conceptual barrier to entry to studying these quantum algorithms and providing new insights remains high, especially for practitioners in the domain where a given problem arises, who may not be familiar with quantum computing beyond the basics. It is thus important to develop tools and methodologies which are accessible to scientists and researchers from different domains, and are as independent of knowing the low-level details of quantum computing as possible, towards enabling easier cross-fertilization of different ideas and techniques. At the same time, it is useful to provide a rigorous general foundation for existing constructions and tools found in the literature, often in a specific context. Thus, a motivating goal of this work is to provide a *design toolkit* of basic results and methodologies which can be used by experts or laymen alike to design, implement, and analyze quantum algorithms. To this end, our results allow for straightforward implementation as computer programs such that Hamiltonian and quantum circuit mappings for many classes of problems may be automatically generated.

In this paper we show a general theory of mappings of Boolean and real functions to diagonal Hamiltonians acting on qubits, and give simple logical rules for the explicit construction of these Hamiltonians which include many common classes of functions such as Boolean formulas and circuits. We also address the question of when such Hamiltonians may or may not be constructed efficiently. We show how our results may be applied to the construction of unitary operators controlled by Boolean predicates, which are used, for example, in several of the QAOA mixing operator constructions of [6]. Our results are general and give a methodical approach to derive many of the mappings in the literature such as those of [6, 8]. We emphasize that our results have applications to quantum algorithms beyond quantum annealing or QAOA, and we discuss a number of examples.

We elaborate on our results, which we summarize in the next section. Consider a function f acting on n bits. We say a Hamiltonian H_f *represents* f if it satisfies

$$H_f|x\rangle = f(x)|x\rangle \tag{1}$$

for each input string $x \in \{0,1\}^n$ with corresponding (i.e., encoded as) computational basis state $|x\rangle$. (We will typically assume f is decidable and moreover can be efficiently evaluated classically; see for example [11] for a discussion of Hamiltonian families that encode the halting problem.) We show how arbitrary n-bit Boolean or real functions are naturally represented as diagonal Hamiltonians given by weighted sums of Pauli Z operators, with terms corresponding to the function's Fourier expansion, as summarized in Theorem 1 below. Such Hamiltonians generalize the Ising model of interacting spin-1/2 particles well-known in physics. Our results rely on the Fourier analysis of Boolean functions, which has a long history of important applications in computer science [12–17] and in particular quantum computing [18–21]. We use our results to derive explicit Hamiltonian representations of a number of basic Boolean predicates shown in Table 1 below. Combining the rules of classical propositional logic with the properties of the Fourier expansion leads to composition rules for constructing Hamiltonians representing conjunctions, disjunctions, exclusive or, and other functions of simpler clauses by combining their Hamiltonian representations in particular ways; see Theorem 2 below. Furthermore, these mappings directly extend to constructing Hamiltonians representing weighted sums of clauses, which are a primary class of Hamiltonians considered in quantum annealing and QAOA, for example, for constraint satisfaction problems.

We also consider the computational complexity of such Hamiltonian constructions, which naturally depends on how the function f is provided as input. Many combinatorial properties of a given function can be "read off" from its Fourier coefficients [22]. This presents an obstruction to computing the Hamiltonian representation for general Boolean functions; we show that computing the identity component of $H_f = \widehat{f}(\emptyset)I + \ldots$, which is given by the first Fourier coefficient $\widehat{f}(\emptyset)$ of f, is as hard as counting the the number of inputs such that $f = 1$, which in general is computationally intractable. For example, if f is a Boolean formula on n variables given in conjunctive normal formula form with a poly(n) size description, i.e., an instance of the satisfiability problem (SAT), then this task is #P-hard and hence it is not believed possible to efficiently

compute $\widehat{f}(\emptyset)$ in general; if such a classical polynomial-time algorithm existed then we would have P=NP. Hence, we cannot efficiently construct explicit Hamiltonian representations of many n-bit Boolean functions, even when such a function may be efficiently evaluated. We apply our results to show a similar result holds for computing the Pauli operator expansion of quantum circuits that compute f in a register.

Nevertheless, there is no such difficulty for *local* Boolean functions f_j where each f_j acts on a constant number of bits. This allows us to efficiently construct Hamiltonians representing *pseudo-Boolean* functions of the form $f(x) = \sum_{j=1}^{m} w_j f_j(x)$, with $w_j \in \mathbb{R}$ and $m = \text{poly}(n)$. Such real functions (and their corresponding Hamiltonians) may be constructed to directly encode optimization problems of interest, or such that their minimum value arguments (ground state eigenvectors) encodes the solution to a corresponding decision problem; similar to, for example, how solving the MAX-SAT problem (finding the maximum possible number of satisfied clauses) also solves SAT. For a pseudo-Boolean function, its Fourier coefficients do not allow its extremal values to be "read off" in the same way and so its Hamiltonian representation can often be computed efficiently. Indeed, this is a common approach to encoding decision problems such as SAT into the framework of quantum annealing [8]. Our results likewise apply to designing penalty term approaches for problems or encodings with hard feasibility constraints which we discuss in Section 3.1.

Table 1: Hamiltonians representing basic Boolean clauses.

$f(x)$	H_f	$f(x)$	H_f
x	$\frac{1}{2}I - \frac{1}{2}Z$	$\overline{x}$	$\frac{1}{2}I + \frac{1}{2}Z$
$x_1 \oplus x_2$	$\frac{1}{2}I - \frac{1}{2}Z_1Z_2$	$\bigoplus_{j=1}^{k} x_j$	$\frac{1}{2}I - \frac{1}{2}Z_1Z_2 \ldots Z_k$
$x_1 \wedge x_2$	$\frac{1}{4}I - \frac{1}{4}(Z_1 + Z_2 - Z_1Z_2)$	$\bigwedge_{j=1}^{k} x_j$	$\frac{1}{2^k}\prod_j (I - Z_j)$
$x_1 \vee x_2$	$\frac{3}{4}I - \frac{1}{4}(Z_1 + Z_2 + Z_1Z_2)$	$\bigvee_{j=1}^{k} x_j$	$I - \frac{1}{2^k}\prod_j (I + Z_j)$
$\overline{x_1 x_2}$	$\frac{3}{4}I + \frac{1}{4}(Z_1 + Z_2 - Z_1Z_2)$	$x_1 \Rightarrow x_2$	$\frac{3}{4}I + \frac{1}{4}(Z_1 - Z_2 + Z_1Z_2)$

1.1 Main Results: A Toolkit for Quantum Optimization Algorithms

We summarize our main results. The details of representing Boolean and real (pseudo-Boolean) functions as Hamiltonians are addressed in Section 2 where we first briefly review and then apply Fourier analysis on the Boolean cube. In Section 3 we outline the application of our results to constructing diagonal operators for optimization applications, including ground state logic approaches and the implementation of diagonal phase unitaries for QAOA. We extend our results to (non-diagonal) controlled unitaries in Section 4, including the setting of operators computing Boolean functions in ancilla registers, which is complementary to our main Hamiltonian approach. Several illustrative examples and remarks are provided throughout. We conclude with a discussion of our results and future research questions in Section 5.

1.1.1 Boolean Functions

Boolean-valued functions on the Boolean cube $\{0,1\}^n$ are uniquely represented as diagonal Hamiltonians as in (1) by a real multilinear polynomial of Pauli Z operators, with terms corresponding to the function's Fourier expansion.

Theorem 1. *For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the unique Hamiltonian on n qubits satisfying $H_f|x\rangle = f(x)|x\rangle$ for each computational basis state $|x\rangle$ is*

$$H_f \;=\; \sum_{S\subset[n]} \widehat{f}(S) \prod_{j\in S} Z_j \;=\; \widehat{f}(\emptyset)I + \sum_{j=1}^{n} \widehat{f}(\{j\})Z_j + \sum_{j<k} \widehat{f}(\{j,k\})Z_jZ_k + \ldots \tag{2}$$

where the Fourier coefficients

$$\widehat{f}(S) = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} f(x)(-1)^{S\cdot x} = \frac{1}{2^n}\text{tr}(H_f \prod_{j\in S} Z_j) \tag{3}$$

satisfy $\widehat{f}(\emptyset) \in [0,1]$, $\widehat{f}(S) \in [-\frac{1}{2}, \frac{1}{2}]$ *for* $S \neq \emptyset$,

$$\sum_{S \subset [n]} \widehat{f}(S) = f(0^n) \tag{4}$$

where 0^n *denotes the input bit string of all* 0*s, and*

$$\sum_{S \subset [n]} \widehat{f}(S)^2 = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) \; = \; \widehat{f}(\emptyset). \tag{5}$$

The proof of the theorem is shown in Sec. 2.1. Here we have used the notation $S \cdot x := \sum_{j \in S} x_j$, $[n] := \{1, 2, \ldots, n\}$, and $Z_j := I \otimes \cdots \otimes I \otimes Z \otimes I \cdots \otimes I$ to denote the Pauli Z operator applied to the jth qubit. We emphasize that the theorem also applies to functions that depend only on a subset of $k < n$ bits, where k may be independent of n. For example, the function $x_1 \oplus x_2 \oplus x_3$ is easily seen to map to the Hamiltonian $\frac{1}{2}I - \frac{1}{2}Z_1Z_2Z_3$. We note that expansions similar to (2) have been considered in the context of implementing diagonal unitaries [23, 24] and analyzing their quantum gate complexity [25], whereas Theorem 1 and our results to follow are more generally applicable. The generalization of Fourier analysis to functions over complex numbers (i.e., quantum circuit amplitudes) has been employed to analyze the complexity of sampling from quantum circuit models such as IQP circuits [21, 26]. We emphasize that in gate model applications, higher order Pauli Z interactions may be implemented efficiently; this is in contrast to quantum annealing applications where interactions must typically be reduced to low-order (quadratic) ones using ancilliary qubits to accommodate physical implementation [8]. See Sec. 3.3 for additional discussion.

Thus, from (5) in we see that computing the Hamiltonian representation (2) of a Boolean function is equivalent to computing its Fourier expansion, and is at least as computationally difficult as computing $\widehat{f}(\emptyset)$.

Corollary 1. *Computing the identity coefficient* $\widehat{f}(\emptyset)$ *of the Hamiltonian* H_f *representing a Boolean satisfiability (SAT) formula* f *(given in conjunctive normal form) is* $\#P$*-hard. Deciding if* $\widehat{f}(\emptyset) = 0$ *is equivalent to deciding if* f *is unsatisfiable, in which case* H_f *reduces to the* 0 *matrix.*

Hence, given a Boolean function f such that counting its number of satisfying inputs is $\#P$-hard, computing the identity coefficient $\widehat{f}(\emptyset)$ of its Hamiltonian representation will be $\#P$-hard also. Such hard counting problems include not only functions corresponding to NP-hard decision problems such as SAT, but also certain functions corresponding to decision problems decidable in polynomial time, such as counting the number of perfect matchings in a bipartite graph; see, e.g., [10]. We emphasize that even if we can compute the value of each Fourier coefficient, a Hamiltonian H_f representing a general Boolean or real function on n bits may require a number of Pauli Z terms that is exponentially large with respect to n; such an example is the logical AND of n variables (see Table 1).

On the other hand, when a Boolean clause f acts only a number of bits $k < n$ that is constant or logarithmically scaling we may always efficiently construct its Hamiltonian representation as the number of nonzero terms in the sum (2) (the *size* of H_f) is in this case at most 2^k. The *degree* of H_f, $\deg(H_f) = \deg(f) = d$, is the maximum locality (number of qubits acted on) of any such term. Bounded-degree Hamiltonians $H_f = \sum_{S \subset [k], |S| \leq d} \widehat{f}(S) \prod_{j \in S} Z_j$, $k \leq n$, are similarly always efficiently representable[1] as we show $\text{size}(H_f) \leq (e/d)^{d-1} k^d + 1$ which is always polynomial in n if $d = O(1)$. For example, bounded-locality constraint satisfaction problems such as MAX-CUT or MAX-ℓ-SAT (up to $\ell = O(\log n)$) are described by sums of local clauses and hence are always efficiently representable as Hamiltonians. We summarize mappings of some important basic clauses in Table 1 above.

1.1.2 Constructing Hamiltonians using Propositional Logic

We show formal rules for combining Hamiltonians representing different Boolean functions to obtain Hamiltonians representing more complicated logical expressions. In particular, we consider the logical negation($\neg$), conjunction ($\wedge$), disjunction ($\vee$), exclusive or ($\oplus$), and implication ($\Rightarrow$) operations, and addition or multiplication by a real number when Boolean functions are embedded as a subset of real-valued functions.

[1] We say a family of functions $\{f_{n,j}\}$ on n bits is *efficiently representable* as the Hamiltonians $\{H_{f_{n,j}}\}$ if $\text{size}(H_{f_{n,j}})$ grows polynomially with n

Theorem 2 (Composition rules)**.** *Let f, g be Boolean functions represented by Hamiltonians H_f, H_g. Then the Hamiltonians representing basic operations on f and g are given by*

- $H_{\neg f} = H_{\overline{f}} = I - H_f$
- $H_{f \wedge g} = H_{fg} = H_f H_g$
- $H_{f \oplus g} = H_f + H_g - 2H_f H_g$
- $H_{f \Rightarrow g} = I - H_f + H_f H_g$
- $H_{f \vee g} = H_f + H_g - H_f H_g$
- $H_{af+bg} = aH_f + bH_g \quad a, b \in \mathbb{R}$.

The proof of theorem is given in Section 2.2. Hamiltonians for a wide variety of functions can be easily constructed using the composition rules and results for basic clauses as in Table 1, in particular Boolean formulas and circuits.

Remark 1. *Theorems 1, 2, and 3 below facilitate straightforward software implementation for generating diagonal Hamiltonians, for example, automating QAOA mappings for constraint satisfaction problems with increasingly general types of constraints. Examples of three variable clauses are generated in Table 2 below. Such Hamiltonians may be useful* intermediate representations *in applications.*

Table 2: Example: Hamiltonians representing Boolean clauses on three variables.

$f(x)$	H_f
$MAJ(x_1, x_2, x_3)$	$\frac{1}{2}I - \frac{1}{4}(Z_1 + Z_2 + Z_3 - Z_1Z_2Z_3)$
$NAE(x_1, x_2, x_3)$	$\frac{3}{4}I - \frac{1}{4}(Z_1Z_2 + Z_1Z_3 + Z_2Z_3)$
$MOD_3(x_1, x_2, x_3)$	$\frac{1}{4}I + \frac{1}{4}(Z_1Z_2 + Z_2Z_3 + Z_1Z_3)$
$1in3(x_1, x_2, x_3)$	$\frac{1}{8}(3I + Z_1 + Z_2 + Z_3 - Z_1Z_2 - Z_2Z_3 - Z_1Z_3 - 3Z_1Z_2Z_3)$

1.1.3 Pseudo-Boolean functions

Consider a real function f on n bits given as a weighted sum of Boolean functions f_j,

$$f(x) = \sum_{j=1}^{m} w_j f_j(x) \qquad w_j \in \mathbb{R},$$

where each f_j acts on a subset of the n bits, and in the applications we consider often $m = \text{poly}(n)$. Objective functions for constraint satisfaction problems, considered for example in QAOA, are often expressed in this form, with each f_j given by a Boolean clause. A different example is the penalty term approach of quantum annealing, where the objective function is augmented with a number of high-weight penalty terms which perform (typically, local) checks that a state is valid; see Sec. 3.1 for a discussion. For such *pseudo-Boolean functions* we have the following result generalizing Thm. 1, which is shown in Section 2.3. Pseudo-Boolean optimization is a rich topic and we refer the reader to [27] for an overview.

Theorem 3. *For an n-bit real function $f : \{0,1\}^n \to \mathbb{R}$ the unique Hamiltonian on n qubits satisfying $H_f|x\rangle = f(x)|x\rangle$ is*

$$H_f = \sum_{S \subset [n]} \widehat{f}(S) \prod_{j \in S} Z_j, \tag{6}$$

with coefficients $\widehat{f}(S) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)(-1)^{S \cdot x} = \frac{1}{2^n} \text{tr}(H_f \prod_{j \in S} Z_j) \in \mathbb{R}$.

In particular, for a pseudo-Boolean function $f(x) = \sum_{j=1}^{m} w_j f_j(x)$, $w_j \in \mathbb{R}$, where the f_j are Boolean functions corresponding to Hamiltonians H_{f_j} as in (2), we have

$$H_f = \sum_{j=1}^{m} w_j H_{f_j} \qquad \textit{and} \qquad \widehat{f}(S) = \sum_j \widehat{f}_j(S) \in \mathbb{R} \tag{7}$$

with $d := \deg(H_f) \leq \max_j \deg(f_j)$ and $\text{size}(H_f) \leq \min\{\sum_j \text{size}(H_{f_j}), (e/d)^{d-1} n^d + 1\}$.

Our results also yield direct cost estimates for quantum simulation of diagonal Hamiltonians, which we elaborate on in Section 3.3, though we emphasize that simulation is not the motivating application our results; nevertheless, it is an important piece of our design toolkit. Indeed, such operators occur for example in QAOA or Grover's algorithm. As the terms in (6) mutually commute, we can *simulate* such an a Hamiltonian, i.e., implement the operator $U = exp(-iH_f t)$ for some fixed $t \in \mathbb{R}$, using $O(\deg(H_f) \cdot \text{size}(H_f))$ many basic quantum gates. Indeed, similar constructions for implementing diagonal unitaries have been previously proposed [23, 24], though more efficient circuits often result by utilizing ancilla qubits such as the operators we consider in Proposition 2 below [28, 29].

The remaining two items of the section demonstrate how our results may be applied to the construction of more general (non-diagonal) Hamiltonians and unitary operators.

1.1.4 Controlled Hamiltonians and Unitaries

In many applications we require controlled Hamiltonian simulations. Such operators are closely related to *block encodings* common in quantum computing [30]. Consider two quantum registers of $k + n$ qubits. Given a Boolean function $f(y)$ acting on k bits and a unitary operator U acting on n qubits, we define the $(k + n)$-qubit f-controlled unitary operator $\Lambda_f(U)$ by its action on computational basis states

$$\Lambda_f(U)|y\rangle|x\rangle = \begin{cases} |y\rangle|x\rangle & f(y) = 0 \\ |y\rangle U|x\rangle & f(y) = 1. \end{cases}$$

Equivalently, as $H_f + H_{\overline{f}} = I$ we have the useful decomposition $\Lambda_f(U) = H_f \otimes U + H_{\overline{f}} \otimes I$.

If U is self-adjoint, then $\Lambda_f(U)$ is also a Hamiltonian. When U is given as a time evolution under a Hamiltonian H for a time t, we have the following.

Proposition 1. *Let f be a Boolean function represented by a k-qubit Hamiltonian H_f, and let H be an arbitrary Hamiltonian acting on n disjoint qubits. Then the $(k + n)$-qubit Hamiltonian*

$$\widetilde{H}_f = H_f \otimes H \tag{8}$$

corresponds to f-controlled evolution under H, i.e., for $t \in \mathbb{R}$ satisfies

$$e^{-i\widetilde{H}_f t} = \Lambda_f(e^{-iHt}). \tag{9}$$

The proof follows from from exponentiating (8) directly, see Section 4.1. We emphasize the Proposition may be applied to generic control functions beyond the AND functions commonly considered in the literature (e.g., multi-controlled Toffoli gates).

Remark 2 (Advanced mixing operators for QAOA)**.** *Proposition 1 may be applied together with the Theorems above to design controlled mixing operators for QAOA mappings of various optimization problems with hard constraints, with the important property of restricting the quantum evolution to the subspace of feasible states [6, 31]. In particular, several mixing operators $\Lambda_f(e^{-iH\alpha})$ proposed in [6] implement evolution under a local mixing Hamiltonian B controlled by a Boolean function f, where the control function checks that the mixing action on a given basis state will maintain feasibility and acts nontrivially only when this is the case. For example, for MaxIndependentSet on a given graph, the transverse-field (bit-flip) mixer is applied for each vertex controlled by the variables corresponding to the its neighbors in the graph; see [6] for details.*

We next consider the special case where the target Hamiltonian H corresponds to a bit flip such that each computational basis state $|y\rangle|0\rangle$ is mapped to $|y\rangle|f(y)\rangle$.

1.1.5 Computing Boolean Functions in a Register

We show how our results may be applied to construct explicit unitary operators which reversibly compute function values in an ancilla qubit register (i.e., implement *oracle queries*), as well as their corresponding Hamiltonians. We remark that a related approach for implementing such operators in a specific gate set has recently appeared [32]. Recall that in the computational basis, the Pauli X operator acts as $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$, i.e., as the *bit-flip*, or NOT, operation.

Proposition 2. *For an n-bit Boolean function f represented by a Hamiltonian H_f, let G_f be the unitary self-adjoint operator on $n+1$ qubits which acts on computational basis states $|x\rangle|a\rangle$ as*

$$G_f|x\rangle|a\rangle = |x\rangle|a \oplus f(x)\rangle \tag{10}$$

where $x \in \{0,1\}^n$ and $a \in \{0,1\}$. Then

$$G_f = \Lambda_f(X) = e^{-i\frac{\pi}{2} H_f \otimes (X-I)}, \tag{11}$$

and $G_f = I$ if and only if f is unsatisfiable.

The proof is given in Section 4.2. As before it remains computationally hard to compute the Fourier expansion of such operators in general.

Corollary 2. *If f is given as a SAT formula in conjunctive normal form, then it is #P-hard to compute the identity coefficient $\widehat{g}(\emptyset) = \mathrm{tr}(G_f)/2^{n+1}$ of G_f, and NP-hard to decide if $\widehat{g}(\emptyset) \neq 1$.*

Equation (11) shows how Hamiltonian simulation may be used to compute a function f in a register. In particular, as the Pauli terms in $H_f \otimes X$ and $H_f \otimes I$ mutually commute, G_f can be implemented with $2 \cdot \mathrm{size}(H_f)$ many multiqubit Pauli rotations with locality up to $\deg(H_f) + 1$.

2 Representing n-bit Functions as Diagonal Qubit Hamiltonians

Many important problems, algorithms, and operators in physics and computer science naturally involve Boolean predicates. Indeed, the relationship between logical propositions and physical observables is foundational in quantum mechanics [33]. For the case of qubits, we show how the natural unique representation of a Boolean function as a Hamiltonian composed of spin operators (Pauli Z matrices) follows from classical Fourier analysis; see [22, 34] for overviews of the subject. We use these tools to extend our results to Hamiltonians representing more general functions built from sums, conjunctions, disjunctions, and other basic combinations of simpler Boolean clauses.

2.1 Boolean Functions

The class of Boolean functions on n bits is defined as $\mathcal{B}_n := \{f : \{0,1\}^n \to \{0,1\}\}$. Taken as the elements of a real vector space, for each n they give a basis for the real functions $\mathcal{R}_n = \{f : \{0,1\}^n \to \mathbb{R}\}$ on n bits. Moreover, $\mathcal{R}_n$ is isomorphic to the vector space of diagonal Hamiltonians acting on n qubits, or, equivalently, the space of $2^n \times 2^n$ diagonal real matrices. Thus, diagonal Hamiltonians naturally encode large classes of functions.

We say a Hamiltonian *represents* a function f if in the computational basis it acts as the corresponding multiplication operator, i.e., it satisfies the 2^n eigenvalue equations

$$\forall x \in \{0,1\}^n \quad H_f|x\rangle = f(x)|x\rangle. \tag{12}$$

On n qubits, this condition specifies H_f uniquely (up to the choice of the computational basis). Equivalently, we may write $H_f = \sum_x f(x)|x\rangle\langle x|$, which for a Boolean function $f \in \mathcal{B}_n$ becomes

$$H_f = \sum_{x:f(x)=1} |x\rangle\langle x|. \tag{13}$$

As Boolean functions satisfy $f^2 = f$ we have $H_f^2 = H_f$, so H_f is a projector[2] of rank $r = \#f := |\{x : f(x) = 1\}| = \sum_x f(x)$. Hence, the Hamiltonian H_f for a Boolean function f is equivalent to the projector onto the subspace spanned by basis vectors $|x\rangle$ such that $f(x) = 1$, and given an f such a projector may be constructed using our results below. Hence, determining if f is satisfiable is equivalent to determining if

[2]Projectors give quantum observables. In particular, for an arbitrary normalized n-qubit state $|\psi\rangle$, the probability p_1 of a computational basis measurement returning a satisfying bit string (i.e., an x such that $f(x) = 1$) is given by $p_1 = \langle\psi|H_f|\psi\rangle$, i.e., is equal to the expected value of repeated measurements of H_f on the state $|\psi\rangle$.

H_f is not identically 0, and determining H_f explicitly in the form of (13) is at least as hard as counting the number of satisfying assignments of f, or equivalently, computing $r = \mathrm{rank}(H_f)$.

We consider the standard computational basis of eigenstates of Pauli Z operators (often written as σ_z), defined by the relations $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. We use $Z_j = I \otimes \ldots I \otimes Z \otimes I \cdots \otimes I$ to denote Z acting on the jth qubit. Products of Z_j over a set of qubits act as

$$\prod_{j \in S} Z_j |x\rangle = \chi_S(x)|x\rangle, \tag{14}$$

where each *parity function* $\chi_S(x) : \{0,1\}^n \to \{-1,+1\}$ gives the parity of the bits of x in the subset $S \subset [n]$, i.e., is $+1$ if and only if the number of bits of x set to 1 is even. Identifying each S with its characteristic vector $S \in \{0,1\}^n$ such that $S \cdot x = \sum_{j \in S} x_j$, we have

$$\chi_S(x) = (-1)^{S \cdot x} = (-1)^{\oplus_{j \in S} x_j}. \tag{15}$$

Thus, each Hamiltonian $Z_S := \prod_{j \in S} Z_j$ represents the function $\chi_S(x)$ in the sense of (12).

The set of parity functions on n bits $\{\chi_S(x) : S \subset [n]\}$ also gives a basis for the real functions $\mathcal{R}_n$. This basis is orthonormal with respect to the inner product defined by

$$\langle f, g \rangle := \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) g(x). \tag{16}$$

Hence, every Boolean function $f \in \mathcal{B}_n$ may be written uniquely as

$$f(x) = \sum_{S \subset [n]} \widehat{f}(S)\, \chi_S(x), \tag{17}$$

called the *Fourier expansion* (or, sometimes, *Walsh* or *Hadamard* expansion [35], or *phase polynomial* [25]) with *Fourier coefficients* given by the inner products of f with the parity functions

$$\widehat{f}(S) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) \chi_S(x) = \langle f, \chi_S \rangle, \tag{18}$$

and satisfying *Parseval's identity*

$$\sum_{S \subset [n]} \widehat{f}(S)^2 = \frac{1}{2^n} \sum_x f(x)^2. \tag{19}$$

The quantity $\sum_{S \neq \emptyset} \widehat{f}(S)^2 =: \mathrm{var}(f)$ is often referred to as the *variance* of f. For $\{0,1\}$-valued functions, we have $f^2 = f$ which implies $\sum_{S \subset [n]} \widehat{f}(S)^2 = \frac{1}{2^n} \sum_x f(x) = \widehat{f}(\emptyset) = \widehat{f}(\emptyset)^2 + \mathrm{var}(f)$.

We refer to the mapping from $f(x)$ to $\widehat{f}(S)$ as the *Fourier transform* of f. The sparsity of f, denoted $\mathrm{spar}(f)$, is the number of non-zero coefficients $\widehat{f}(S)$. When $\mathrm{spar}(f) = \mathrm{poly}(n)$ we refer to f as *polynomially sparse*; in particular, polynomially sparse functions yield Hamiltonians of size $\mathrm{poly}(n)$. The *degree* of f, denoted $\deg(f)$, is defined to be the largest $|S|$ such that $\widehat{f}(S)$ is nonzero. Note that if f depends on only $k \leq n$ variables, then $\deg(f) \leq k$.

We summarize our results on the representation of Boolean functions in Theorem 1 above.

Proof of Theorem 1. From the identification of χ_S with $Z_S = \bigotimes_{j \in S} Z_j = \prod_{j \in S} Z_j$, from (17) we see that each Boolean function f is represented as a diagonal Hamiltonian by a linear combination of tensor products of Z_j operators. As all Pauli operators and their tensor products are traceless except for the identity operator, we have $\widehat{f}(S) = \mathrm{tr}(H_f \prod_{j \in S} Z_j)/2^n$where [3] The results (4) and (5) follow from Parseval's identity (19) using $f^2 = f$. Recall we define the degree (sometimes called the *Pauli weight*) of such a Hamiltonian H_f, $\deg(H_f)$, to be the largest number of qubits acted on by any term in this sum, and the size, $\mathrm{size}(H_f)$, to be the number of (nonzero) terms. Clearly, we have $\deg(H_f) = \deg(f) =: d$, and simple counting and the bound $\binom{n}{d} \leq n^d (e/d)^d / e$ gives $\mathrm{size}(H_f) = \mathrm{spar}(f) \leq d\binom{n}{d} \leq (e/d)^{d-1} n^d + 1$. □

[3] $\mathrm{tr}(H)$ denotes the *trace* of the matrix H, i.e., the (basis-independent) sum of its diagonal elements,

We emphasize that the Hamiltonian coefficients $\widehat{f}(S)$ depend only on the function values $f(x)$, and are independent of *how* such a function may be represented as input (e.g., formula, circuit, truth table, etc.). Many typical compact representations of Boolean functions as computational input such as Boolean formulas or Boolean circuits can be directly transformed to Hamiltonians using the composition rules of Theorem 2 which we derive below.

Remark 3. *The identification of Boolean functions as real polynomials allows application of Theorem 1 to different domains or targets. Changing the target from* $\{0,1\}$ *to* $\{1,-1\}$ *for a given* $f(x)$ *corresponds to the function* $g(x) = 2f(x) - 1$ *with Hamiltonian coefficients* $\widehat{g}(\emptyset) = 1 - 2\widehat{f}(\emptyset)$ *and* $\widehat{g}(S) = -2\widehat{f}(S)$ *for* $S \neq \emptyset$*, so this can change* $\mathrm{size}(H_f)$ *by at most* 1*. (In this case there are several differences from the* $\{0,1\}$ *approach; for example, Parseval's identity trivially becomes* $\sum_{S\subset[n]} \widehat{f}(S)^2 = 1$*.)*

Similarly, indeed, the Fourier expansion (17) itself corresponds to changing the domain from $\{0,1\}^n$ *to* $\{1,-1\}^n$ *(i.e., the polynomial obtained replacing each* Z_j *in (6) by the variable* $z_j = (-1)^{x_i} \in \{-1,1\}$*).*

On the other hand, Theorem 1 shows that computing the Hamiltonian representation of a Boolean function is as hard as computing its number of satisfying assignments, which is believed to be a computationally intractable problem in general [10]. We illustrate this explicitly in Corollary 1 above where we show computing $\widehat{f}(\emptyset)$ can be #P-hard. Moreover, arbitrary Boolean functions may have size (sparsity) exponential in n, in which case, even if we know somehow its Hamiltonian representation, we cannot implement or simulate this Hamiltonian efficiently (with respect to n) with the usual direct approaches.

As explained, Hamiltonians representing pseudo-Boolean functions often avoid these difficulties; for example, constraint satisfaction problems with objective function given as the sum of a number of *local* clauses each clause acting on at most $k = O(\log n)$ bits (e.g., Max-k-Sat), and so the Hamiltonians representing each clause have degree $O(\log n)$ and size $O(\mathrm{poly}(n))$, and hence can be efficiently constructed. Applying the well-known results of [15, Thm. 1 & 2] for such functions to Theorem 1 immediately gives the following useful Hamiltonian degree bounds.

Corollary 3. *For a function* $f \in \mathcal{B}_n$ *that depends only on* $k \leq n$ *variables, represented as a Hamiltonian* H_f *acting on* n *qubits, the degree of* H_f *satisfies*

$$k \geq D(f) \geq \deg(H_f) \geq \log_2 k - O(\log\log k), \tag{20}$$

where $D(f)$ *is the decision tree complexity of* f *and* $D(f) = O(\mathrm{poly}(\deg(H_f)))$*.*

We emphasize that further results from the literature concerning the Fourier analysis of Boolean functions may be similarly adapted to obtain properties of corresponding Hamiltonians. In particular additional useful details of the Fourier coefficients may be found in [22, 25, 34].

2.2 Basic Clauses and Logical Composition Rules

Boolean functions arise in many different applications, but are often given in or easily reduced to a *normal form*. For example, SAT formulas are given in conjunctive normal form. Many other normal forms exist such as disjunctive, algebraic ($\oplus$), minterm or maxterm, etc. [36]. Note that while logically equivalent, the different forms may be quite different for computational purposes. For each form there corresponds a notion of size (which directly relates to the number of bits needed to describe a function in such a form).

The laws of classical propositional logic allow for convenient manipulation of a given Boolean expression between logically equivalent forms. Hence, given Hamiltonians representing basic Boolean functions, it is useful to have a methodical way to combine them to in order to construct new Hamiltonians representing their logical conjunctions (AND), disjunctions (OR), etc. This approach often allows for easier construction of such a Hamiltonian than by working with the Fourier expansion directly. First, consider Hamiltonians representing basic functions and variables. The trivial functions $f = 1$ and $f = 0$ (resp. *true* and *false*) are represented by the 2^n-dimensional Hamiltonians $H_1 = I$ and $H_0 = 0$, respectively. Using the identity $(-1)^x = 1 - 2x$, we represent the jth variable x_j, considered as a multiplication operator, as the Hamiltonian

$$H_{x_j} \;=\; I^{\otimes j-1} \otimes |1_j\rangle\langle 1_j| \otimes I^{\otimes n-j} \;=\; \frac{1}{2}I - \frac{1}{2}Z_j, \tag{21}$$

which acts according to the value of the jth bit as $H_{x_j}|x\rangle = x_j|x\rangle$. Similarly, the logical negation of the jth variable is represented as $H_{\overline{x}_j} = I/2 + Z_j/2$. For clarity we will avoid writing tensor factors of identity operators explicitly when there is no ambiguity, and for convenience we sometimes write x_j to mean the Hamiltonian H_{x_j}, and likewise for other basic functions such as $\overline{x}_j$.

Applying the laws of propositional logical leads to the composition rules of Theorem 2.

Proof of Thm. 2. The logical values 1 and 0 (i.e., *true* and *false*) are represented as the identity matrix I and the zero matrix, respectively. Each result follows from the natural embedding of $f, g \in \mathcal{B}_n$ into $\mathcal{R}_n$, the real vector space of real functions on n bits. From linearity of the Fourier transform, we immediately have $H_{af+bg} = aH_f + bH_g$ for $a, b \in \mathbb{R}$. Using standard identities, the Boolean operations $(\cdot, \vee, \oplus, \dots)$ on f, g can be translated into $(\cdot, +)$ formulas, i.e., linear combinations of f and g. Linearity then gives the resulting Hamiltonian in terms of H_f and H_g. Explicitly, for the complement of a function $\overline{f}$, as $\overline{f} = 1 - f$, we have $H_{\overline{f}} = I - H_f$. Similarly, the logical identities $f \wedge g = fg$, $f \vee g = f + g - fg$, $f \oplus g = f + g - 2fg$, and $f \Rightarrow g = \overline{f} + fg$, respectively, imply the remaining results of the theorem. □

We summarize the Hamiltonian representations of several basic Boolean functions in Table 1 above, which are easily derived from Theorem 1. Applying the laws of Theorem 2 we may derive Hamiltonians representing more complicated Boolean formulas than those of Table 1, such as expressions with arbitrary numbers of variables, mixed types of clauses, or given as Boolean circuits. Some typical examples of Boolean functions on 3 variables are the Majority (MAJ), Not-All-Equal (NAE), and 1-in-3 functions, which behave as their names indicate. The Mod_3 function is 1 when the sum $x_1 + x_2 + x_3$ is divisible by 3, and satisfies $\mathrm{Mod}_3 = \overline{NAE}$. We show the Hamiltonians representing these functions in Table 2, which may be derived using either the composition rules of Theorem 2 or the Fourier expansion approach of Theorem 1. For example, H_{1in3} follows applying Thm. 2 with the identity $1in3(x_1, x_2, x_3) = x_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3$.

The rules of Theorem 2 may be applied *recursively* to construct Hamiltonians representing more complicated Boolean functions, corresponding e.g. to parentheses in logical formulas, or wires in Boolean circuits. For example, the Hamiltonian representing the Boolean clause $f \vee g \vee h = f \vee (g \vee h)$ is given by $H_{f \vee g \vee h} = H_f + H_{g \vee h} - H_f H_{g \vee h}$, which simplifies to

$$H_{f \vee g \vee h} = H_f + H_g + H_h - H_f H_g - H_f H_h - H_g H_h + H_f H_g H_h.$$

Another example is the Majority function which satisfies

$$H_{MAJ(f,g,h)} = -2H_f H_g H_h + H_f H_g + H_f H_h + H_g H_h.$$

Together, the rules of Theorem 2 and Table 1 show how to construct Hamiltonians representing functions given as arbitrary Boolean algebra $(\wedge, \vee)$ or Boolean ring $(\cdot, \oplus)$ elements, which are functionally complete in the sense of representing all possible Boolean functions [36].

2.3 Pseudo-Boolean Functions and Constraint Satisfaction Problems

Real functions on n bits are similarly represented as diagonal Hamiltonians via their Fourier expansion. Every such function $f \in \mathcal{R}_n$ may be expanded (non-uniquely) as a weighted sum of Boolean functions, possibly of exponential size. By linearity of the Fourier transform, the Hamiltonian H_f is given precisely by the corresponding weighted sum of the Hamiltonians representing the Boolean functions. Moreover, the Hamiltonian H_f is unique, so different expansions of f as sums of Boolean functions must all result in the same H_f.

The Fourier coefficients are again given by the inner product (16) with the parity functions χ_S,

$$\widehat{f}(S) = \langle f, \chi_S \rangle = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)\chi_S(x), \tag{22}$$

and satisfy Parseval's identity as stated in (19). We are particularly interested in *pseudo-Boolean functions* given as a weighted sum of logical clauses

$$f(x) = \sum_{j=1}^{m} w_j f_j(x), \tag{23}$$

where $f_j \in \mathcal{B}_n$ and $w_j \in \mathbb{R}$. Note that we do not deal explicitly with how the real numbers w_j are represented and stored; for many applications they are bounded rational numbers and this issue is relatively minor; see, e.g., the constructions in [6,8]. Indeed, in a constraint satisfaction problem, typically all $w_j = 1$ and hence $f(x)$ gives the number of satisfied clauses (constraints).

We have the following result which extends the previous results for Boolean functions.

Proof of Theorem 3. By the linearity of the Fourier expansion and Theorem 1 we have that an n-bit real function $f : \{0,1\}^n \to \mathbb{R}$ is represented as the Hamiltonian $H_f = \sum_{S\subset[n]} \widehat{f}(S) \prod_{j\in S} Z_j$ with $\widehat{f}(S) = \langle f, \chi_s\rangle = \frac{1}{2^n}\mathrm{tr}(H_f \prod_{j\in S} Z_j) \in \mathbb{R}$. For pseudo-Boolean functions $f = \sum_{j=1}^m w_j f_j$ the bounds $d := \deg(H_f) \leq \max_j \deg(f_j)$ and $\mathrm{size}(H_f) \leq \min\{\sum_j \mathrm{size}(H_{f_j}), (e/d)^{d-1}n^d + 1\}$ follow from simple counting as in the proof of Theorem 1. □

Thus the results of Theorems 1 and 2 for Boolean functions also apply to the construction of Hamiltonians representing real functions, though with several important distinctions. Various subclasses of so-called pseudo-Boolean functions with particular attributes (e.g., submodularity) are important in applications and their properties may be further studied; see [27] for an overview.

Remark 4. *In contrast to Theorem 1, for a constraint satisfaction problem $f = \sum_{j=1}^m f_j$, with $f_j \in \mathcal{B}_n$, applying Parseval's identity (19) we have*

$$\sum_{S\subset[n]} \widehat{f}(S)^2 = \mathbf{E}[f] + 2\sum_{i<j}\langle f_i, f_j\rangle = \widehat{f}(\emptyset) + 2\sum_{i<j}\mathbf{E}[f_i \wedge f_j] \geq \mathbf{E}[f],$$

where $\mathbf{E}[f] := \frac{1}{2^n}\sum_{x\in\{0,1\}^n} f(x)$ gives the expected value of $f(x)$ over the uniform distribution. In particular, $\sum_{S\subset[n]} \widehat{f}(S)^2 = \mathbf{E}[f]$ if and only if $\langle f_i, f_j\rangle = 0$ for all i, j. If there does exist an i, j such that $\langle f_i, f_j\rangle = 0$, then the conjunction of the clauses is unsatisfiable, i.e. $\wedge_j f_j = 0$.

3 Applications to Constructing Diagonal Operators

Here we overview four immediate applications of our results to diagonal operators and problem encodings: constrained optimization, and ground state logic, simulating diagonal Hamiltonians, and quadratic optimization.

3.1 Constrained Optimization

In constrained optimization problems, we seek an optimal solution subject to satisfying a set of *feasibility* constraints. These constraints may arise as part of the problem itself or from its encoding [6,8]. In quantum annealing, a common approach to dealing with problem constraints is to augment the Hamiltonian that represents the objective function to be minimized with additional diagonal Hamiltonian terms that penalize (i.e., shift the eigenvalues of) states outside of the feasible subspace [3,8].

Suppose we are given a constrained real function $f(x)$ to minimize, with a set of Boolean hard constraint functions g_j, $j = 1, \ldots, \ell$, such that at least one of the $g_j(x) = 1$ if x is an infeasible solution, and so $\sum_j g_j(x) = 0$ if and only if x is feasible. (Here, we may always redefine f as to take a convenient value on infeasible states.) We may construct the augmented problem Hamiltonian as

$$H_p = H_f + \sum_{j=1}^{\ell} w_j H_{g_j},$$

where the Hamiltonians H_f and H_{g_j} represent f and the g_j as in Theorems 3 and 1, respectively. The w_j are positive weights which may be selected appropriately such that infeasible basis states are eigenvectors of H_p with eigenvalues shifted away from those of f (e.g., $w_j > \max(x)f(x)$), and feasible states are eigenvectors with eigenvalues $f(x)$. Hence, the ground state subspace of H_p is spanned by states representing optimal feasible problem solutions. Theorems 2, 1, and 3 may be applied to the functions g_j to construct the penalty terms H_{g_j} just as for the cost term H_f.

Thus, our results may also be applied to explicitly construct problem mappings with penalty terms for constrained problems. See [8] for a number of specific problem mappings; we emphasize our approach may be applied directly to problems not considered therein. Furthermore, similar ideas apply to related approaches for constrained optimization such as Lagrange multipliers [37].

An alternative approach to constrained optimization is to map the hard constraint functions to diagonal Hamiltonians, which may be used to design mixing Hamiltonian that preserves the subspace of feasible states. This approach is proposed in constraint-preserving generalizations of quantum annealing [38, 39] and QAOA [6, 31], and can offer advantages over penalty-term approaches; see [6, 31, 38, 39] for details and example problem mappings.

3.2 Ground State Boolean Logic

With a universal quantum gate-model computer, a single (additional) control bit suffices for all efficiently computable control functions [40]. Indeed, in principle we can always compute a Boolean function $f(x)$ in a control register by constructing a unitary operator $U_f : |x\rangle|a\rangle \to |x\rangle|a \oplus f(x)\rangle$, as we consider in Section 4.2. Nevertheless, in certain models or applications it is desirable to have a purely Hamiltonian implementation.

A different approach to computing a Boolean function $f \in \mathcal{B}_n$ in a register is to encode its input-output pairs as the ground state subspace of a Hamiltonian H, referred to as *ground state Boolean logic*; see, e.g., [41–43]. There are different ways to encode a function as the ground state subspace, as in there is freedom in how the Hamiltonian acts on invalid computational basis states. For example, the function $AND(x, y) = xy$ can be encoded with the subspace $span\{|x\rangle|y\rangle|xy\rangle\} = span\{|000\rangle, |010\rangle, |100\rangle, |111\rangle\}$, which corresponds to the ground state subspace of a number of Hamiltonians. To construct such a Hamiltonian, the penalty term approach of the previous section could be used to penalize the invalid states. An alternative construction that takes advantage of the Hamiltonian representation H_f is to directly implement the Boolean function $g \in \mathcal{B}_{n+1}$ that satisfies $g(x, y) = 0$ if and only if $y = f(x)$, or equivalently $g(x, y) = \overline{f(x) \oplus y}$. Applying Theorems 1 and 2 to this function and simplifying gives the following result.

Proposition 3. *Let $f \in \mathcal{B}_n$ be represented by the Hamiltonian H_f as in Theorem 1, and $x_a := \frac{1}{2}I - \frac{1}{2}Z_a$ where the ancilla qubit is labelled a. Then the $(n+1)$-qubit Hamiltonian H_g*

$$H_g = I \otimes x_a + H_f \otimes Z_a \tag{24}$$

represents the Boolean function $g \in \mathcal{B}_{n+1}$ which satisfies $g(x, y) = 0$ if and only if $y = f(x)$, and has ground state subspace given by

$$span\{|x\rangle|f(x)\rangle : x \in \{0, 1\}^n\}.$$

Simpler Hamiltonians with the same ground state subspace may be found for specific classes of Boolean functions f; see, e.g., [44]. An advantage of the construction (24) is that it applies generally.

3.3 Simulating Diagonal Hamiltonians

Here we explain how Theorems 1 to 3 may be straightforwardly applied to yield quantum circuits implementing time evolution under diagonal Hamiltonians, i.e., diagonal unitaries, which is a fairly well-studied problem with several related approaches proposed in the literature.

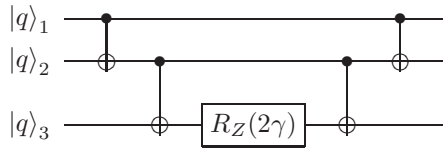

Fig. 1: Quantum circuit performing the operation $U = exp(-i\gamma Z_1 Z_2 Z_3)$ on three qubits labeled 1, 2, and 3. The middle operator is a Z-rotation gate, and the other gates are controlled-NOT (CNOT) gates with a black circle indicating the control qubit and cross indicating the target. By similar circuits, $U = exp(-i\gamma Z_1 Z_2 \dots Z_\ell)$ can be implemented with $2(\ell - 1)$ CNOT gates and one R_Z gate. Different circuit compilations are possible, including compilation to different gate sets.

In many applications we desire to *simulate* a Hamiltonian H for some time $\gamma \in \mathbb{R}$, i.e., implement exactly or approximately the unitary operator $U(\gamma) = e^{-i\gamma H}$. When H is diagonal, efficient quantum circuits may be obtained using Fourier analysis [23–25] or other approaches. Consider the simulation of a Hamiltonian H_f representing a real or Boolean function f. It is well known that if f can be efficiently computed classically, and if ancilla qubits are available, then the Hamiltonian H_f can be simulated efficiently by computing f in a scratchpad register and performing a sequence of controlled rotations; see, e.g., [29]. These methods typically avoid computing the Fourier expansion of f explicitly. On the other hand, there exist applications where an explicit Hamiltonian-based implementation is desirable, such as quantum annealing, or cases where we wish to minimize the need for ancilla qubits, such as, for example, near-term implementations. Efficient circuits simulating products of Pauli Z operators are well-known [28, 29], as shown in Figure 1. As Pauli Z terms mutually commute, circuits simulating individual terms in the Hamiltonians (2) or (6) can be applied in any sequence as to simulate their sum. Thus, when $\text{size}(H_f) = O(\text{poly}(n))$ we can always simulate H_f efficiently in this way. We summarize this observation in the following.

Corollary 4. *A Hamiltonian H_f representing a Boolean or real function f as in (2) or (6) can be simulated, i.e., the operation $exp(-i\gamma H_f)$ implemented, with n qubits and $O(\deg(H_f) \cdot size(H_f))$ basic quantum gates. In particular, Hamiltonians H_f with bounded maximum degree $d := \deg(H_f) = O(1)$ can be simulated with $O(n^d)$ basic gates. Ancilla qubits are not necessary in either case.*

Here, by *basic quantum gates* we mean the set of CNOT and single qubit rotation gates, which is a standard universal set [28, 40]. We remark that the Hamiltonian simulation considered in the corollary is exact in the sense that if each of the basic gates is implemented exactly, then so is $exp(-i\gamma H_f)$. The approximation of quantum gates and operators is an important topic but we do not deal with it here; see, e.g., [40].

Example 1. *[Application to Grover's algorithm] For a Boolean function f, simulating H_f for time π gives the standard oracle query for Grover's algorithm [40]*

$$e^{-i\pi H_f}|x\rangle = (-1)^{f(x)}|x\rangle. \tag{25}$$

Hence, when H_f is known explicitly and $size(H_f) = \text{poly}(n)$, we can efficiently construct and implement the operator $(-1)^{f(x)}$ using quantum circuits for simulating H_f using only CNOT and R_Z gates, without any necessary ancilla qubits.

3.4 Quadratic Unconstrained Binary Optimization

A general and important class of pseudo-Boolean optimization problems are *quadratic unconstrained binary optimization* (QUBO) problems [3], where we seek to maximize or minimize a degree-two pseudo-Boolean function

$$f(x) = a + \sum_{j=1}^{n} c_j x_j + \sum_{j<k} d_{jk} x_j x_k, \tag{26}$$

with $a, c_j, d_{jk} \in \mathbb{R}$ and $x_j \in \{0, 1\}$. Indeed, this is the class of problems (ideally) implementable on current quantum annealing devices such as, for example, D-WAVE machines, where the qubit interactions are themselves quadratic [3, 44]. The QUBO class also contains many problems which at first sight are not quadratic, via polynomial reductions which often require extra variables; indeed, the natural QUBO decision problem is NP-complete [45]. Note that $\overline{x}_j = 1 - x_j$, so (26) is without loss of generality. Applying our above results gives the following.

Corollary 5. *The QUBO objective function (26) maps to a Hamiltonian given as a quadratic sum of Pauli Z operators, with $size(H_f) \leq 1 + n/2 + n^2/2$. Explicitly, we have*

$$H_f = (a + c + d)I - \frac{1}{2}\sum_{j=1}^{n}(c_j + d_j)Z_j + \frac{1}{4}\sum_{j<k} d_{jk} Z_j Z_k, \tag{27}$$

where we have defined $c = \frac{1}{2}\sum_{j=1}^{n} c_j, d = \frac{1}{4}\sum_{j<k} d_{jk}$, and $d_j = \frac{1}{2}\sum_{k:k\neq j} d_{jk}$ with $d_{jk} = d_{kj}$.

Moreover, we can simulate H_f, i.e., implement the phase operator $U_p(t) = e^{-itH_f}$, using at most n many R_Z rotation gates and $\binom{n}{2}$ many R_{ZZ} gates.

The QUBO problem is closely related to the Ising model of interacting spins from physics [46].

Example 2 (Interacting Ising spins)**.** *Consider the related classical ISING decision problem of determining whether the ground state energy (lowest eigenvalue) of an Ising model (degree two) diagonal Hamiltonian*

$$H = a_0 I + \sum_j a_j Z_j + \sum_{j<k} a_{jk} Z_j Z_k$$

is at most a given constant. As the number of terms is size$(H) = O(n^2)$, we can efficiently check the energy $H(x)$ of each candidate ground state $|x\rangle$, $x \in \{0,1\}^n$, so the problem is in NP. On the other hand, from Theorem 2, we see that the NP-complete problem MAX-2-SAT maps to a degree-two Hamiltonian of this form, with solution encoded in the ground state energy of $-H$. Thus, from our results it trivially follows that ISING is NP-complete. Similar arguments can be used to show NP-completeness with restricted coefficients values (e.g., antiferromagnetic) or with restricted interaction topologies such as planar graphs [47].

4 Applications to non-diagonal operators

Here we consider the application of our results beyond strictly diagonal operators, in particular to constructing controlled unitaries (quantum gates) common in quantum algorithms.

Just as we have seen that diagonal Hamiltonians correspond to linear combinations of Pauli Z operators, tensor products of arbitrary single-qubit Pauli matrices give a basis for the vector space of n-qubit Hamiltonians. Indeed, a general Hamiltonian H may be expanded as a sum of Pauli matrices as

$$H = a_0 I + \sum_{j=1}^{n} \sum_{\sigma=X,Y,Z} a_{j\sigma} \sigma_j + \sum_{j\neq k} \sum_{\sigma=X,Y,Z} \sum_{\lambda=X,Y,Z} a_{jk\sigma\lambda} \sigma_j \lambda_k + \ldots, \tag{28}$$

with real coefficients $a_\alpha \in \mathbb{R}$. (For general linear operators on qubits the same formula holds but with $a_\alpha \in \mathbb{C}$, see e.g. [48].) The coefficients are easily shown to satisfy

$$a_\alpha = \frac{1}{2^n} \mathrm{tr}(\alpha H), \tag{29}$$

for each of the 4^n Pauli terms $\alpha = \sigma_1 \sigma_2 \ldots \sigma_n$, $\sigma_j \in \{I, X_j, Y_j, Z_j\}$, which are orthonormal with respect to the Hilbert-Schmidt inner product $\langle \alpha, \beta \rangle := \frac{1}{2^n} \mathrm{tr}(\alpha^\dagger \beta)$ that generalizes (16).

4.1 Controlled Unitaries and Hamiltonians

Many quantum algorithms require controlled Hamiltonian evolutions. For example, in quantum phase estimation (QPE) [40], we require transformations on $(1+n)$-qubit basis states of the form

$$|0\rangle|x\rangle \to |0\rangle|x\rangle, \qquad |1\rangle|x\rangle \to |1\rangle e^{-iHt}|x\rangle,$$

for various values $t = 1, 2, 4, \ldots$. Consider such a transformation with fixed t. Labeling the first register (control qubit) a, the overall unitary may be written as

$$\Lambda_{x_a}(e^{-iHt}) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes e^{-iHt}. \tag{30}$$

Recall the notation $\Lambda_{x_a}(e^{-iHt})$ indicates the unitary e^{-iHt} controlled by the classical function x_a. We obtain the Hamiltonian corresponding to this transformation by writing $\Lambda_{x_a}(e^{-iHt}) = e^{-i\widetilde{H}t}$, which gives

$$\widetilde{H} = |1\rangle\langle 1| \otimes H = x_a \otimes H = \frac{1}{2} I \otimes H - \frac{1}{2} Z_a \otimes H. \tag{31}$$

Recall that for simplicity we sometimes write a function f in place of its Hamiltonian representation H_f as we have done here for the function $f(x) = x_a$. Note that the control qubit is assumed precomputed here; its value may or may not depend on x.

More generally, consider Hamiltonian evolution controlled by a Boolean function $g \in \mathcal{B}_k$ acting on a k-qubit ancilla register. In this case we seek to affect the unitary transformation on $(k+n)$-qubit basis states

$$|y\rangle|x\rangle \to |y\rangle|x\rangle \qquad \text{if } g(y) = 0,$$
$$|y\rangle|x\rangle \to |y\rangle e^{-iHt}|x\rangle \qquad \text{if } g(y) = 1,$$

which gives the overall unitary

$$\Lambda_g(e^{-iHt}) = \sum_{y:g(y)=0} |y\rangle\langle y| \otimes I + \sum_{y:g(y)=1} |y\rangle\langle y| \otimes e^{-iHt} = H_{\overline{g}} \otimes I + H_g \otimes e^{-iHt}, \tag{32}$$

corresponding to evolution under the Hamiltonian

$$\widetilde{H}_g = \sum_{y:g(y)=1} |y\rangle\langle y| \otimes H = H_g \otimes H. \tag{33}$$

These results have been summarized in Proposition 1 above. Note that if the Hamiltonian $H = H_f$ represents a Boolean function f, then (33) represents the conjunction of f and g, i.e., we have $\widetilde{H}_g = H_g \otimes H_f = H_{g \wedge f}$.

Moreover, for an arbitrary Hamiltonian H, if we can implement the ancilla controlled operator $\Lambda_{x_a}(e^{-iHt})$ for sufficiently many values of $t \in \mathbb{R}$, then it is straightforward to implement a variable-time controlled Hamiltonian simulation operator $\Lambda_{\tau,H}$, which acts on basis states as

$$\Lambda_{\tau,H}|\tau\rangle|x\rangle = |\tau\rangle e^{-iH\tau}|x\rangle. \tag{34}$$

For example, if τ was encoded in binary, then Λ_H could be implemented by applying the operators $\Lambda_{x_{a_0}}(e^{-iH2^0})$, $\Lambda_{x_{a_1}}(e^{-iH2^1})$, ..., $\Lambda_{x_{a_j}}(e^{-iH2^j})$, ... in sequence controlled over each bit τ_j in the first register; see, e.g., [40].

Finally, we remark on the condition that the control function and target unitaries act on disjoint sets of qubits, and the relation to spin creation and annihilation operators.

Example 3 (Creation/annihilation operators and interacting fermions)**.** *Consider the operator Xx which acts as $Xx|0\rangle = 0$ and $Xx|1\rangle = |0\rangle$. Clearly $Xx = X(I-Z)/2 = X/2 + iY/2$ is not self-adjoint. Indeed, Xx is equivalently the well-known spin annihilation operator $b = |0\rangle\langle 1|$. The spin creation operator $b^\dagger$ is similarly given as $b^\dagger = (Xx)^\dagger = xX = X\overline{x} = |1\rangle\langle 0|$, and the spin number (occupation) operator is $b^\dagger b = x = (I-Z)/2$. Thus, the product of two Hamiltonians H_g and H acting nontrivially on overlapping qubits is not guaranteed to yield a self-adjoint operator as in (33).*

For systems of fermions in the second quantized (occupation number) representation [49], Hamiltonians consist of polynomials of fermionic creation and annihilation operators $a^\dagger$ and a, satisfying the canonical anticommutation relation algebra $\{a_j, a_k\} = \{a_j^\dagger, a_k^\dagger\} = 0$ and $\{a_j, a_k^\dagger\} = \delta_{jk}$. It is easily shown that these relations are satisfied if we use spin operators and the parity functions χ_S, $S \subset [n]$ of (15) to represent the fermionic operators as $a_j = \chi_{\{1,2,\ldots,j-1\}} b_j$. Hence, from above we may represent these operators as $a_j = Z_1 Z_2 \ldots Z_{j-1}(X_j + iY_j)/2$ and $a_j^\dagger = Z_1 Z_2 \ldots Z_{j-1}(X_j - iY_j)/2$ which reproduces the well-known Jordan-Wigner transform. This representation is used in many applications, such as quantum algorithms for quantum chemistry, though alternative mappings with different tradeoffs are known [50].

4.2 Computing Functions in Registers

Here we consider the special case of computing a Boolean function in a register and show a similar complexity result as for the diagonal Hamiltonian case. As an illustrative example we briefly consider the connection of our results to quantum query complexity and Grover's algorithm.

Suppose for a Boolean function f we have a unitary operator G_f that acts on each $(n+1)$-qubit basis state $|x\rangle|a\rangle$, $a \in \{0,1\}$ as

$$G_f|x\rangle|a\rangle = |x\rangle|a \oplus f(x)\rangle. \tag{35}$$

If we let the function f be arbitrary and unknown, then each application of G_f (considered a black-box) is called an *oracle query* for f. Operators G_f as in (35) are called *bit queries*. Note that G_f may often be

derived from a reversible classical circuit for computing f, but we consider it abstractly here. (The query (25) for Grover's algorithm is often referred to as a *phase query*.)

We show that G_f also induces a diagonal representation of H_f, and hence faces the same computational difficulties for the Boolean case. Observe that the diagonal Hamiltonian

$$H'_f := G_f x_a G_f = \frac{1}{2}I - \frac{1}{2}G_f Z_a G_f = x_a + H_f Z_a \tag{36}$$

acts on computational basis states as

$$H'_f|x\rangle|a\rangle = (f(x) \oplus a)|x\rangle|a\rangle, \tag{37}$$

so we identify $H'_f = H_{f\oplus a}$. Hence when $a = 0$, H'_f gives an $n+1$ qubit representation of f.

Proof of Prop. 2. The first statement follows from applying Prop. 1 with H_f as given from Thm. 1 and $H = X_a$. For the second statement, we use (28), (29), and (37) to expand G_f as a sum of Pauli terms acting on n qubits $G_f = (1 - \widehat{f}(\emptyset))I + \widehat{f}(\emptyset)X_a + \ldots$, where none of the terms to the right are proportional to I. Hence the identity coefficient $\widehat{g}(\emptyset) = \mathrm{tr}(G_f)/2^{n+1} = 1 - \widehat{f}(\emptyset)$ gives (one minus) the fraction of satisfying assignments of f. □

We conclude the section with a pedagogic example application to oracle models.

Example 4 (Phase kickback and function queries)**.** *Recall Example 1. For an arbitrary Boolean function f, a single application of the bit-query oracle G_f suffices to simulate $(-1)^f$ as*

$$G_f|x\rangle|-\rangle_a = (-1)^{f(x)}|x\rangle|-\rangle_a, \tag{38}$$

using a single ancilla qubit prepared in the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = H|1\rangle$ and where H denotes the Hadamard (single qubit quantum Fourier transform) gate [40], known as phase kickback.

Next consider the controlled-phase-query oracle $\Lambda_{x_a}((-1)^f) = \Lambda_{x_a}(e^{-i\pi H_f}) = (-1)^{f\wedge x_a}$ derived from (30). Observe that if we can query $\Lambda_{x_a}((-1)^f)$ then we can implement G_f using a single-bit quantum phase estimation, which requires two Hadamard gates applied to an ancilla qubit and a single $\Lambda_{x_a}((-1)^f)$ query as

$$H_a\, \Lambda_{x_a}((-1)^f)\, H_a|x\rangle|0\rangle_a = |x\rangle|0 \oplus f(x)\rangle_a = G_f|x\rangle|0\rangle_a. \tag{39}$$

Similarly, two bit queries can simulate $\Lambda_{x_a}((-1)^f)$ using an ancilla qubit $|0\rangle_b$ to store $f(x)$ as

$$(I \otimes G_f)\; R_{Z_a}(-\tfrac{\pi}{2}) R_{Z_b}(-\tfrac{\pi}{2}) R_{Z_a Z_b}(\tfrac{\pi}{2})\; (I \otimes G_f)\; |x_a\rangle_a|x\rangle|0\rangle_b = c'(-1)^{f(x)\wedge x_a}|x_a\rangle_a|x\rangle|0\rangle_b, \tag{40}$$

where the constant c' is an unimportant global phase. (Here the three rotations on the left of (40) are derived from simulating the Hamiltonian representing the function $x_a \wedge x_b$ for time π as to implement the operator $(-1)^{x_a\wedge x_b}$, and the second application of G_f uncomputes the ancilla qubit.) Hence, we easily see that the oracles G_f and $\Lambda_{x_a}(e^{-i\pi H_f}) = (-1)^{f(x)\wedge x_a}$ are computationally equivalent, and both are at least as powerful as the phase oracle $(-1)^f$. The results of this paper may be similarly applied to the study of complexity and reductions between further classes of oracles.

5 Discussion and Future Work

We have shown explicit rules and results for constructing Hamiltonians representing Boolean and pseudo-Boolean functions, including important classes of objective functions for combinatorial optimization. Applications include quantum gate-model and quantum annealing approaches to optimization, the simulation of diagonal Hamiltonians, and the construction of penalty or mixer Hamiltonians for problems with hard constraints. Moreover, we have shown our results entail quantum circuit constructions for controlled unitaries, in particular, those that reversibly compute a Boolean function in an ancilla register. The goal of these results is to give a toolkit which can be used generally towards the design and implementation of a wide range of quantum approaches for optimization, for related applications such as machine learning [51],

and beyond. Our results give a unified view of existing problem mappings in the literature, such as those of [6, 8].

There are a variety of enticing applications and extensions of our results, and we briefly outline several additional directions. We emphasize that Fourier analysis is generally a very rich topic in computer science, mathematics, and physics, in addition to its many applications in quantum computing. A promising research direction is to further apply these tools, in particular more advanced ideas from the Fourier analysis of Boolean functions as applied in classical computer science, to the design and analysis of quantum algorithms. As mentioned, we leave a detailed analysis of general Hamiltonians acting on qubits or qudits as a topic of future work. A next step is to further classify *unfaithful* Hamiltonian representations of Boolean functions, where n variables are encoded in $n' > n$ qubits, which in particular is an important paradigm for embedding problems on physical quantum annealing hardware, and, more generally, is related to the theory of quantum error correcting codes. Moreover, our results may have useful applications to quantum statistical mechanics, where many important Hamiltonians are given as linear combination of Pauli operators, such as the Ising or quantum XY models [47, 52]. Furthermore, exploring connections to quantum complexity theory may be fruitful, such as Hamiltonian complexity [53–55], or the computational power of restricted classes of quantum circuits [26,56,57]. Finally, it of interest whether techniques from quantum computing and quantum information can shed further insight on important open problems in classical computer science [17, 58].

Acknowledgments

We thank Al Aho for support and guidance, and the members of the Quantum AI Lab for providing helpful comments and suggestions. This work was initiated thanks to the support of the USRA Feynman Quantum Computing Academy summer internship program and NASA Ames Research Center, and further developed while at Columbia University. S.H. was additionally supported by NASA Academic Mission Services, Contract No. NNA16BD14C. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose not withstanding any copyright annotation thereon.

References

[1] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Phys. Rev. E*, vol. 58, no. 5, p. 5355, 1998.

[2] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum computation by adiabatic evolution," *arXiv preprint quant-ph/0001106*, 2000.

[3] C. C. McGeoch, "Adiabatic quantum computation and quantum annealing: Theory and practice," *Synthesis Lectures on Quantum Computing*, vol. 5, no. 2, pp. 1–93, 2014.

[4] T. Hogg and D. Portnov, "Quantum optimization," *Information Sciences*, vol. 128, no. 3-4, pp. 181–197, 2000.

[5] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[6] S. Hadfield, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, 2019.

[7] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, 2014.

[8] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, no. 5, pp. 1–15, 2014.

[9] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[10] S. Arora and B. Barak, *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[11] M. A. Nielsen, "Computable functions, quantum measurements, and quantum dynamics," *Physical Review Letters*, vol. 79, no. 15, p. 2915, 1997.

[12] J. Kahn, G. Kalai, and N. Linial, "The influence of variables on Boolean functions," in *Proc. 29th IEEE Symposium on Foundations of Computer Science*, pp. 68–80, IEEE, 1988.

[13] N. Linial, Y. Mansour, and N. Nisan, "Constant depth circuits, Fourier transform, and learnability," *Journal of the ACM (JACM)*, vol. 40, no. 3, pp. 607–620, 1993.

[14] R. Beigel, "The polynomial method in circuit complexity," in *Proc. 8th Structure in Complexity Theory Conference*, pp. 82–95, IEEE, 1993.

[15] N. Nisan and M. Szegedy, "On the degree of Boolean functions as real polynomials," *Computational complexity*, vol. 4, no. 4, pp. 301–313, 1994.

[16] P. L. Hammer and S. Rudeanu, *Boolean Methods in Operations Research and Related Areas*, vol. 7. Springer Science & Business Media, 2012.

[17] Y. Gu and X.-L. Qi, "Majorana fermions and the sensitivity conjecture," *arXiv preprint arXiv:1908.06322*, 2019.

[18] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. De Wolf, "Quantum lower bounds by polynomials," *Journal of the ACM (JACM)*, vol. 48, no. 4, pp. 778–797, 2001.

[19] A. Ambainis, "Polynomial degree vs. quantum query complexity," *Journal of Computer and System Sciences*, vol. 72, no. 2, pp. 220–238, 2006.

[20] A. Montanaro and T. J. Osborne, "Quantum Boolean functions," *arXiv preprint arXiv:0810.2435*, 2008.

[21] S. Boixo, V. N. Smelyanskiy, and H. Neven, "Fourier analysis of sampling from noisy chaotic quantum circuits," *arXiv preprint arXiv:1708.01875*, 2017.

[22] R. O'Donnell, *Analysis of Boolean functions*. Cambridge University Press, 2014.

[23] N. Schuch and J. Siewert, "Programmable networks for quantum algorithms," *Physical review letters*, vol. 91, no. 2, p. 027902, 2003.

[24] J. Welch, D. Greenbaum, S. Mostame, and A. Aspuru-Guzik, "Efficient quantum circuits for diagonal unitaries without ancillas," *New Journal of Physics*, vol. 16, no. 3, p. 033040, 2014.

[25] M. Amy, P. Azimzadeh, and M. Mosca, "On the controlled-not complexity of controlled-not–phase circuits," *Quantum Science and Technology*, vol. 4, no. 1, p. 015002, 2018.

[26] M. J. Bremner, A. Montanaro, and D. J. Shepherd, "Achieving quantum supremacy with sparse and noisy commuting quantum computations," *Quantum*, vol. 1, p. 8, 2017.

[27] E. Boros and P. L. Hammer, "Pseudo-Boolean optimization," *Discrete applied mathematics*, vol. 123, no. 1, pp. 155–225, 2002.

[28] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical review A*, vol. 52, no. 5, p. 3457, 1995.

[29] A. M. Childs, *Quantum information processing in continuous time*. PhD thesis, Massachusetts Institute of Technology, 2004.

[30] S. Chakraborty, A. Gilyén, and S. Jeffery, "The power of block-encoded matrix powers: improved regression techniques via faster hamiltonian simulation," *arXiv preprint arXiv:1804.01973*, 2018.

[31] S. Hadfield, Z. Wang, E. G. Rieffel, B. O'Gorman, D. Venturelli, and R. Biswas, "Quantum approximate optimization with hard and soft constraints," in *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, PMES'17, (New York, NY, USA), p. 15–21, Association for Computing Machinery, 2017.

[32] M. Soeken and M. Roetteler, "Quantum circuits for functionally controlled not gates," *arXiv preprint arXiv:2005.12310*, 2020.

[33] G. Birkhoff and J. Von Neumann, "The logic of quantum mechanics," *Annals of mathematics*, pp. 823–843, 1936.

[34] R. De Wolf, "A brief introduction to Fourier analysis on the Boolean cube.," *Theory of Computing, Graduate Surveys*, vol. 1, pp. 1–20, 2008.

[35] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*, vol. 16. Elsevier, 1977.

[36] S. Givant and P. Halmos, *Introduction to Boolean Algebras*. Springer Science & Business Media, 2008.

[37] M. Ohzeki, "Breaking limitation of quantum annealer in solving optimization problems under constraints," *Scientific reports*, vol. 10, no. 1, pp. 1–12, 2020.

[38] I. Hen and M. S. Sarandy, "Driver Hamiltonians for constrained optimization in quantum annealing," *Phys. Rev. A*, vol. 93, no. 6, p. 062312, 2016.

[39] I. Hen and F. M. Spedalieri, "Quantum annealing for constrained optimization," *Phys. Rev. Appl.*, vol. 5, no. 3, p. 034007, 2016.

[40] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge UK: Cambridge University Press, 2000.

[41] J. Biamonte, "Nonperturbative k-body to two-body commuting conversion Hamiltonians and embedding problem instances into Ising spins," *Phys. Rev. A*, vol. 77, no. 5, p. 052331, 2008.

[42] I. J. Crosson, D. Bacon, and K. R. Brown, "Making classical ground-state spin computing fault-tolerant," *Phys. Rev. E*, vol. 82, no. 3, p. 031106, 2010.

[43] J. D. Whitfield, M. Faccin, and J. Biamonte, "Ground-state spin logic," *EPL (Europhysics Letters)*, vol. 99, no. 5, p. 57004, 2012.

[44] Z. Bian, F. Chudak, W. G. Macready, and G. Rose, "The Ising model: teaching an old problem new tricks," tech. rep., D-Wave Systems, 2010.

[45] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP–Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[46] H. Nishimori, *Statistical physics of spin glasses and information processing: an introduction*, vol. 111. Clarendon Press, 2001.

[47] F. Barahona, "On the computational complexity of ising spin glass models," *Journal of Physics A: Mathematical and General*, vol. 15, no. 10, p. 3241, 1982.

[48] P. Woit, *Quantum Theory, Groups and Representations: An Introduction*. Springer, 2017.

[49] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. Dover, 1996.

[50] J. T. Seeley, M. J. Richard, and P. J. Love, "The Bravyi-Kitaev transformation for quantum computation of electronic structure," *The Journal of chemical physics*, vol. 137, no. 22, p. 224109, 2012.

[51] G. Verdon, M. Broughton, and J. Biamonte, "A quantum algorithm to train neural networks using low-depth circuits," *arXiv preprint arXiv:1712.05304*, 2017.

[52] E. Lieb, T. Schultz, and D. Mattis, "Two soluble models of an antiferromagnetic chain," in *Condensed Matter Physics and Exactly Soluble Models*, pp. 543–601, Springer, 2004.

[53] J. Kempe, A. Kitaev, and O. Regev, "The complexity of the local Hamiltonian problem," *SIAM J. Comput.*, vol. 35, no. 5, pp. 1070–1097, 2006.

[54] S. Gharibian, Y. Huang, Z. Landau, S. W. Shin, *et al.*, "Quantum Hamiltonian complexity," *Foundations and Trends in Theoretical Computer Science*, vol. 10, no. 3, pp. 159–282, 2015.

[55] T. S. Cubitt, A. Montanaro, and S. Piddock, "Universal quantum hamiltonians," *Proceedings of the National Academy of Sciences*, vol. 115, no. 38, pp. 9497–9502, 2018.

[56] E. Knill and R. Laflamme, "Power of one bit of quantum information," *Physical Review Letters*, vol. 81, no. 25, p. 5672, 1998.

[57] S. Bravyi, D. Gosset, and R. König, "Quantum advantage with shallow circuits," *Science*, vol. 362, no. 6412, pp. 308–311, 2018.

[58] Y. Filmus, H. Hatami, S. Heilman, E. Mossel, R. O'Donnell, S. Sachdeva, A. Wan, and K. Wimmer, "Real analysis in computer science: A collection of open problems," *Preprint available at https://simons. berkeley. edu/sites/default/files/openprobsmerged.pdf*, 2014.