Design of General Purpose Minimal-Auxiliary Ising Machines

Isaac K. Martin[†]
Department of Mathematics
University of Texas at Austin
Austin, Texas
Email: ikmartin@utexas.edu

Andrew G. Moore[†]
Department of Mathematics
University of Texas at Austin
Austin, Texas
Email: agmoore@utexas.edu

John T. Daly
Advanced Computing Systems
Laboratory for Physical Sciences
Catonsville, Maryland
Email: jtdaly3@lps.umd.edu

Jess J. Meyer
Advanced Computing Systems
Laboratory for Physical Sciences
Catonsville, Maryland
Email: jess@lps.umd.edu

Teresa M. Ranadive
Advanced Computing Systems
Laboratory for Physical Sciences
Catonsville, Maryland
Email: tranadive@lps.umd.edu

Abstract—**Ising machines are a form of quantum-inspired processing-in-memory computer which has shown great promise for overcoming the limitations of traditional computing paradigms while operating at a fraction of the energy use. The process of designing Ising machines is known as the reverse Ising problem. Unfortunately, this problem is in general computationally intractable: it is a nonconvex mixed-integer linear programming problem which cannot be naively brute-forced except in the simplest cases due to exponential scaling of runtime with number of spins. We prove new theoretical results which allow us to reduce the search space to one with quadratic scaling. We utilize this theory to develop general purpose algorithmic solutions to the reverse Ising problem. In particular, we demonstrate Ising formulations of 3-bit and 4-bit integer multiplication which use fewer total spins than previously known methods by a factor of more than three. Our results increase the practicality of implementing such circuits on modern Ising hardware, where spins are at a premium.**

1. Introduction

The limitations of the von Neumann model of computing become clearer with each passing year. Therefore, it is important to explore both potential unconventional theoretical models of future computing and the hardware techniques which could enable their implementation. This paper will focus on the theoretical design of general-purpose Ising machines by attempting to specify quadratic Hamiltonians with arbitrary prescribed ground states and minimal auxiliary spins. This model is closely related to reversible Boltzmann machines, adiabatic quantum computing, and classical Hopfield artificial neural networks [9]; in fact, Ising Hamiltonians can be used to create algorithms for quantum computers [1].

Most work in the field of Ising algorithms has been focused on reformulating NP-complete and NP-hard optimization problems as the minimization of Ising Hamiltonians, including the travelling salesman, max-cut, and knapsack problems [14]. Indeed, the max-cut problem has become a standard benchmark for physical Ising-type hardware. We are instead interested in creating Ising circuits which implement arbitrary logical functionality, especially integer multiplication. Previously, it has been demonstrated that arbitrary logic gates and ripple-carry addition circuits can be encoded as quadratic Ising Hamiltonians with minimal auxiliary spins [26] and hence that Ising-type systems are capable of universal computation [13]. However, the practical design of multiplication circuits turns out to be much more difficult than addition.

The minimization of Ising Hamiltonians can be achieved with a wide variety of hardware, including optical coherent Ising Machines [16], simulation on D-Wave quantum annealers [1, 5], digital FPGA implementations [21], trapped ions [17], and analog oscillators with sub-harmonic injection locking [10, 25]. Each implementation technology has its own set of advantages and restrictions, but as all are still in the early stages of research, we are not concerned with working to specific architectural requirements, but rather with establishing general theory. As such, we will not attempt to minimize interaction strength dynamic range, ground state energy gap, or graph connectivity. The Ising systems discussed in this paper are zero-temperature infinite range classical spin glasses with real number interaction strengths.

The contributions of this paper are two-fold. The first is a new mathematical theory of Ising circuits which adds clarity to the problem, reveals connections to Boolean analysis and machine learning, and yields powerful theoretical results that dramatically reduce the complexity of designing Ising circuits. The second is a pair of nondeterministic algorithms which exploit our mathematical theory; combined with a number of notable optimizations, they can be applied to

---

[†]. These authors contributed equally.

solve arbitrary reverse Ising problems. Section 2 details the development of our theory and the proofs of our core results. As a showcase of its utility, we include a new constructive algorithmic proof of the universality of Ising systems which follows immediately from results in pseudo-Boolean optimization. Section 3 deals with the implementation of our algorithms, the design of an optimized Mehrotra predictor-corrector method, and a number of optimizations derived from the structure and symmetry of the reverse Ising problem. In Section 4 we apply our theory and algorithms to the 3-bit and 4-bit integer multiplication circuits. Our solutions reduce the total number of spins required to implement these circuits by a factor of more than 3 compared to previous work.[1]

2. Theory

Here we establish a theoretical framework of Ising circuits and the reverse Ising problem. Ising spins are objects with a binary state space Σ, variously referred to as 'up and down' or '1 and -1' or '1 and 0'. Different formulations are convenient for different situations, and when it is relevant we will explicitly state whether we are using $\Sigma = \{-1, 1\}$ or $\Sigma = \{0, 1\}$ convention. Note that though all qualitative results are interchangeable under a change of variables.

2.1. The Reverse Ising Problem

Throughout this paper we replace the lattice Λ in the traditional Ising model with an arbitrary finite set $X \subset \mathbb{N}$ and define Σ^X to be the set of all functions $X \longrightarrow \Sigma$. Such a function σ assigns an Ising spin state to each element of X, and hence Σ^X the set of all possible Ising states or configurations of X. For this reason we call Σ^X the *spin space* of X. See [8] for historical conventions. When $A \subseteq X$ is a subset, for every spin state $\sigma \in \Sigma^X$ the restriction $\sigma|_A$ is an element of Σ^A and thus represents the state of the subset A.

Since we are primarily interested in the spin space Σ^X, it might seem more natural to instead take X to be some positive integer and Σ^X to be the collection of vectors in $\mathbb{R}^{|X|}$ valued in $\Sigma = \{0, 1\}$ or $\{\pm 1\}$. This convention better reflects the implementation of our algorithms but introduces the need to carefully track coordinates. We default to the coordinate-free approach as it streamlines notation and is far more convenient for mathematical formalism. The two conventions are nonetheless entirely equivalent and one can freely move between them by replacing X with its cardinality $|X|$, choosing coordinates on $\mathbb{R}^{|X|}$, and identifying each spin state $\sigma \in \Sigma^X$ with a corresponding vertex of the hypercube.

1. The implementation of integer multiplication on Ising hardware turns out to be quite difficult, and hence serves as a good testing ground for our methods. Integer multiplication is also a fundamental binary operation, and is hence of interest to those wishing to implement logic operations on Ising hardware. The upcoming publication [18] addresses integer multiplication more specifically.

2.1.1. Circuits. A **circuit** is a triple (N, M, f) where $N, M \subseteq X$ are disjoint subsets of X called the collections of *input* and *output* spins respectively, and f is the logic function $f : \Sigma^N \longrightarrow \Sigma^M$ mapping spin states of N to spin states of M. The collection $A = X \setminus (N \cup M)$ of spins which are neither input nor output spins is called the set of *auxiliary spins*. The spinspace Σ^X can now be canonically identified with $\Sigma^N \times \Sigma^M \times \Sigma^A$ by identifying each spin state $\sigma \in \Sigma^X$ with $(\sigma|_N, \sigma|_M, \sigma|_A)$.

For a fixed $\sigma \in \Sigma^X$ it is sometimes useful to consider the collection $\mathcal{L}_\sigma = \{\sigma' \in \Sigma^X \mid \sigma'|_N = \sigma|_N\}$ of all spin states matching σ in the input component. We call $\mathcal{L}_\sigma$ the σ-*input level* of the circuit (N, M, f), or when the choice of circuit (N, M, f) is clear, simply the *input level of* σ.

2.1.2. Ising Systems. An **Ising system** is a pair (X, H) where X is a finite set whose elements are called *spins* and $H : \Sigma^X \longrightarrow \mathbb{R}$ is a *quadratic pseudo-Boolean polynomial* (see Section 2.2.1) called the *Hamiltonian of X*. Classically, the linear coefficients of the Hamiltonian are called *local biases* while the quadratic coefficients are called *coupling coefficients*. The likelihood of observing an Ising system in a state $\sigma \in \Sigma^X$ at a temperature T is given by the **configuration probability** or **Boltzmann probability**

$$P_\beta(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z_\beta} \qquad (1)$$

where $\beta = (k_B T)^{-1}$ is the inverse temperature, k_B is the Boltzmann constant, and the normalization constant Z_β is the partition function $Z_\beta = \sum_{\sigma \in \Sigma^X} e^{-\beta H(\sigma)}$. In the low-temperature limit, the probability that the system will be in its ground state is 1.

There are many ways to write the Hamiltonian H of an Ising system (X, H). We make quick note of the two most useful conventions here.

1) For distinct $i, j \in X$ denote by h_i the local bias of i and by J_{ij} the coupling strength of i and j. For a spin state $\sigma \in \Sigma^X$ we can then write

$$H(\sigma) = \sum_{i \in X} h_i \sigma(i) + \sum_{i < j} J_{ij} \sigma(i) \sigma(j). \qquad (2)$$

2) Interpreting $\sigma \in \Sigma^X$ as a vector with entries in ± 1 and denoting by $\sigma \otimes \sigma$ the outer product, define the *virtual spin* of σ to be σ concatenated with its $\sigma \otimes \sigma$: $v(\sigma) = (\sigma, \sigma \otimes \sigma)$. We can then write $H(\sigma)$ as the inner product

$$H(\sigma) = \langle u, v(\sigma) \rangle, \qquad (3)$$

where u is the *coefficient vector* of H.[2]

2. As stated, these two conventions for the Hamiltonian are exactly equivalent if we allow a constant term added in Equation 2. However, the outer product $\sigma \otimes \sigma$ is a symmetric matrix with nonzero diagonal, and thus appears to introduce extra terms in the Hamiltonian. These go away once you expand the inner product, in which case $J_{ij} = u_{ij} + u_{ji}$. In practice, we only consider the upper triangular portion of $\sigma \otimes \sigma$ to reduce dimensionality. We therefore think of v as an embedding $\Sigma^X \hookrightarrow \Sigma^{|X| + \binom{|X|}{2}}$. This latter spin space is not physical, hence the term "virtual spin".

2.1.3. The Reverse Ising Problem. Given a circuit (N, M, f) with indexing set X, we wish to design an Ising system (X, H) whose behavior realizes (N, M, f) with high probability. We will now make this problem precise.

We assume the input spin states can be fixed while the rest of the system evolves freely according to Ising dynamics[3]. For an input state $\sigma \in \Sigma^N$ we therefore wish to maximize the probability that the output spins of the Ising system are found in the correct state $f(\sigma)$. In the low temperature limit $(\beta \gg 1)$ this is a simple optimization problem: find a Hamiltonian $H : \Sigma^N \times \Sigma^M \times \Sigma^A \longrightarrow \mathbb{R}$ such that for each input state $\sigma \in \Sigma^N$, whenever $\eta \in \mathcal{L}_\sigma$ minimizes the Hamiltonian H on the input level $\mathcal{L}_\sigma$, $\eta|_M$ is equal to $f(\sigma)$:

$$\eta \in \arg\min_{\omega \in \mathcal{L}(\sigma)} H(\omega) \implies \eta|_M = f(\sigma). \tag{4}$$

The circuit data prescribe only a preferred output component for every input state. If we additionally have an auxiliary function $g : \Sigma^N \longrightarrow \Sigma^A$ prescribing a preferred auxiliary component, then this becomes a linear programming problem in the coefficients of H: Given a circuit (N, M, f), find an Ising system (X, H) such that for every input state $\sigma \in \Sigma^N$, the following constraints are satisfied:

$$H(\sigma, \omega, \eta) > H(\sigma, f(\sigma), g(\sigma)) \tag{5}$$

for all $\eta \in \Sigma^A$ and all incorrect output states $f(\sigma) \neq \omega \in \Sigma^M$. These constraints can be written as the vector inequality

$$Bu > 0 \tag{6}$$

where u is the coefficient vector of the Hamiltonian H and B is the *constraint matrix* whose rows are given by the differences $v(\sigma, \omega, \eta) - v(\sigma, f(\sigma), g(\sigma))$.

We call the problem of finding such an auxiliary function g along with a suitable Hamiltonian the ***Reverse Ising Problem***, and if the above constraints are satisfied, we say (X, H) **solves** (N, M, f). The data of (N, M, f) and (X, H) together is called an *Ising circuit.* It is important to note that because finding H is simply a linear programming problem, the determination of g is the source of nearly all the difficulty. We say that a choice of g is *feasible* if the linear problem given in Equation 5 is feasible. Our primary goal is to find feasible choices of g for a given circuit (N, M, f)—furthermore, we would like A to be a small as possible, since spin sites are expensive. In other words, we are seeking minimal-size auxiliary maps that make feasible the realization of a given function f as an Ising Hamiltonian.

2.2. General Observations

It is not obvious at a glance whether or not the reverse Ising problem is solvable for all circuits. In fact, it is always

solvable. In this section, we cover established results from Boolean analysis which give us a constructive algorithmic proof of this result. We also discuss the relationship of Ising circuits to Hopfield networks and Support Vector Machines (SVMs), and resolve some apparent difficulties resulting from the comparison.

2.2.1. Universality of Quadratic Ising Systems. We will work with $\Sigma = \{0, 1\}$. A *pseudo-Boolean function* is any function of the form $f : \Sigma^n \longrightarrow \mathbb{R}$. It is a basic result in Boolean analysis that every pseudo-Boolean functions f has a unique representation as a multilinear polynomial

$$\sum_{H \subseteq \{1, \ldots, n\}} c_H \prod_{i \in H} x_i \tag{7}$$

where the c_H are the Hadamard coefficients of f [6]. We may therefore regard any pseudo-Boolean function as a multilinear polynomial. Each degree n monomial refers to an n-local spin interaction, so a physically realizable Ising Hamiltonian is a quadratic pseudo-Boolean polynomial. It is easy to see that every circuit (N, M, f) is solvable with a higher degree Hamiltonian polynomial $H : \Sigma^{N+M} \longrightarrow \mathbb{R}$ and zero auxiliaries by letting $H(\sigma, \eta) = d(\eta, f(\sigma))$ where d is the Hamming distance (note that $H(\sigma, \eta) \geq 0$ and $H(\sigma, \eta) = 0 \iff \eta = f(\sigma)$). However, while some work has been done on higher-order spin interactions in Ising circuit design [3], higher-order spin interactions are generally regarded as unphysical and/or infeasible to implement. Fortunately, *quadratization* techniques exist to convert higher-order polynomial minimization problems into quadratic unconstrained binary optimization (QUBO) problems.

A **quadratization** of a pseudo-Boolean polynomial $p(\vec{x})$ is a quadratic pseudo-Boolean polynomial $q(\vec{x}, \vec{a})$ such that

$$\min_{\vec{a} \in \Sigma^A} q(\vec{x}, \vec{a}) = p(\vec{x}). \tag{8}$$

A wide variety of quadratization techniques exist [11], but the best general purpose algorithm in terms of minimizing the number of auxiliary values A which are added is the Rosenberg reduction algorithm, which substitutes a product $x_i x_j$ with a new auxiliary variable a by adding the penalty term[4] $P = C(xy - 2ax - 2ay + 3a)$ [24, 4]. It is easy to check that $P \geq 0$ and $P = 0 \iff a = xy$, so substitution is guaranteed at global minima. In particular, since this process always terminates on finite polynomials, combining the previous two remarks by applying Rosenberg quadratization to the Hadamard transform of the Hamiltonian $H(\sigma, \eta) = d(\eta, f(\sigma))$ leads to the following guarantee:

Proposition 2.1. The reverse Ising problem is solvable for any circuit.

Example 2.2. It follows from an interesting quadratization result of Boros, Crama, and Rodríguez-Heck [7] that an Ising circuit which evaluates the parity of n input bits has an elegant closed form solution. Let $p(x)$ be the parity check

---

3. There are multiple ways to accomplish this depending on the specific hardware implementation. For instance, the input local biases can be made to dominate the other terms of the Hamiltonian or the inputs can be made to be ferromagnetic pairs.

4. $C > \sum |c_H|$ is sufficient.

function on $\{0,1\}^n$ which returns 1 if $\sum x_i$ is even and zero otherwise. Note that $\arg \min_{y \in \{0,1\}} p(x, y) = p(x)$, so $p(x, y)$ is a valid higher degree Hamiltonian for the parity check circuit. Following the paper's result, this Hamiltonian can be quadratized with $\ell := \lceil \log(n + 1) \rceil - 1$ auxiliaries; explicitly, $H(x, y, a)$ is

$$\left(\sum_{i=1}^n x_i + y + 2^{\ell+1} - \sum_{i=1}^{\ell} 2^i a_i - (n \mod 2) \right)^2 \quad (9)$$

In combination with the main result of this paper, this implies that parity-checking auxiliary bits can be glued to a circuit at relatively low cost. This has important applications for the implementation of LDPC (Low Density Parity Check) encoding and decoding circuits as Ising circuits [5].

2.2.2. Single-Output Ising Circuits are SVMs. Intuition is greatly aided by concrete analysis of simple cases. Let $\Sigma = \{\pm 1\}$. Consider the case that $M = 1$, i.e. a circuit with a single output bit. We can express any quadratic Hamiltonian as $H(\vec{x}, y) = y\tilde{H}(\vec{x}) + R(\vec{x})$ where $\tilde{H}$ is linear and R is homogeneous and quadratic. If we insist that wrong states have energy at least 1 higher than correct states (as is usually done in practice for numerical convenience; see Equation 16), then the constraint set for the reverse Ising problem with zero auxiliaries is

$$H(\vec{x}, f(x)) + 1 \le H(\vec{x}, -f(x)) \qquad \forall \vec{x} \in \Sigma^N \quad (10)$$

Note then that

$$H(\vec{x}, f(x)) + 1 \le H(\vec{x}, -f(x)) \quad (11)$$
$$\iff f(x)\tilde{H}(\vec{x}) + 1 \le -f(x)\tilde{H}(\vec{x}) \quad (12)$$
$$\iff f(x)(-2\tilde{H}(\vec{x})) \ge 1 \quad (13)$$

Since $-2\tilde{H}(x)$ is linear, it is expressible as $\langle w, x \rangle - b$ for $w \in \mathbb{R}^N, b \in \mathbb{R}$. This shows that the constraint set for the reverse Ising problem is precisely the constraints for the hard-margin SVM problem, where getting the correct output is understood as a binary classification problem. Since Σ^N embeds into $\mathbb{R}^N$ as the vertices of the N-dimensional hypercube, it follows that:

Proposition 2.3. A circuit $(N, 1, f)$ is solvable without auxiliaries if and only if f is a threshold function on the N-hypercube.

Each boolean function $f : \Sigma^d \longrightarrow \Sigma$ can be thought of as a labeling of the vertices of an d-dimensional hypercube embedded in $\mathbb{R}^d$. The false set of f is $f^{-1}(0)$ and likewise $f^{-1}(1)$ is the true set of f. We say that f is a *d-dimensional threshold function* if $f^{-1}(0)$ and $f^{-1}(0)$ are linearly separable; that is, when there exists some hyperplane $L_{\mathbf{w}, b} = \{\mathbf{x} \in \mathbb{R}^N \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$ such that $f^{-1}(0)$ is the set of vertices below the plane and $f^{-1}(1)$ is the set of vertices above the plane. These are well studied and have been counted up to $d = 9$, see [12] for an overview and [2] for more recent results. We discuss the analog of soft-margin SVMs and various applications of this result in Section 3.1.

2.3. Augmented Constraints

Solutions to the reverse Ising problem naïvely happen in two steps:

1) The auxiliary problem: find an appropriate size for A and an auxiliary function $g : \Sigma^N \longrightarrow \Sigma^A$. This is a nonlinear nonconvex mixed-integer constrained optimization problem.
2) The linear problem: solve the linear programming problem (Equation 5).

Early attempts at general algorithmic solutions to the reverse Ising problem attempt to make iterative improvements to an initial choice of auxiliary function $g : \Sigma^N \longrightarrow \Sigma^A$ by measuring the feasibility of g using some heuristic. A quick analysis reveals this to be unsuitable for all but the simplest circuits. Finding a feasible auxiliary function g involves searching through the space of all possible auxiliary functions (a set of size $2^{A \cdot 2^N}$), and the lack of convexity in the auxiliary problem means that feasibility heuristics are of limited use. Worse, feasibility heuristics generally require a pass through the linear problem (see Section 3 for details). A quick check of Equation 5 reveals the number of constraints scales exponentially in A—there are precisely $2^N \cdot (2^{M+A} - 2^A)$ constraints, each of length[5] $M + A + \binom{N+M+A}{2} - \binom{N}{2} \sim \mathcal{O}(N + M + A)^2$. Therefore the difficulty of the linear problem for assessing a candidate g with respected to a fixed circuit (N, M, f) grows like $\mathcal{O}(2^A A^2)$ as A increases. The exponential scaling makes finding g for all but the smallest circuits practically impossible. A more sophisticated approach is needed.

There are two obvious avenues for improvement: (1) cut down on the size of the auxiliary search space and (2) reduce the cost of the linear problem. Theorem 2.4 provides sizable improvements on both of these fronts by drastically reducing the search space of possible auxiliary functions and eliminating the exponential scaling in A, thus reducing the complexity of the linear problem to $\mathcal{O}(A^2)$ with respect to a fixed circuit. The key insight is to allow the auxiliary function g to depend on both Σ^N and Σ^M rather than only Σ^N.

Theorem 2.4. *Let (N, M, f) be a circuit. There exists an Ising system which solves this circuit if and only if there is a function $g : \Sigma^N \times \Sigma^M \longrightarrow \Sigma^A$ such that*

(a) The circuit $(N \cup M, A, g)$ is solvable by an Ising system with Hamiltonian R with the following additional property:

$$R(\sigma, \omega, g(\sigma, \omega)) \ge R(\sigma, f(\sigma), g(\sigma, f(\sigma))) \quad (14)$$

*for all input states σ and output states ω. Equation 14 is the **weak neutralizability condition**. If we instead have equality then it is the **strong neutralizability condition**. If such an Ising system (X,R) exists, we correspondingly*

---

5. The virtual spin has $N + M + A + \binom{N+M+A}{2}$ components, but the columns corresponding to combinations of spins in N are always zero in the constraint matrix, and can therefore be removed.

say that g is solvable weakly neutralizable or solvable strongly neutralizable.

*(b) There is an Ising system (X, H) which satisfies g-**augmented constraints**:*

$$H(\sigma, \omega, g(\sigma, \omega)) > H(\sigma, f(\sigma), g(\sigma, f(\sigma))). \quad (15)$$

*We call (X, H) the **base system** and the circuit (N, M, f) the **base circuit**. We call the system (X, R) the **auxiliary system** and the circuit $(N \cup M, A, g)$ the **auxiliary circuit**.*

The proof can be found in the appendix of the eprint version of this paper [15].

Remark 2.5. Weak neutralizability depends on the logic of the circuit (N, M, f); an auxiliary functions g may be weakly neutralizable for some choices of f but not for others. Strong neutralizability does not depend on f.

Remark 2.6. At first glance it seems we have made the situation worse – Theorem 2.4 splits the task of finding a single Ising system which solves (N, M, f) into the task of finding two Ising systems satisfying distinct constraint sets. The advantage becomes clear given the following two observations:

1) **Improvements to g can be made iteratively.** It is easy to check that if both $(N \cup M, A_1, g_1)$ and $(N \cup M, A_2, g_2)$ are solvable weakly neutralizable auxiliary circuits, then they can be "glued" together to form a new circuit $(N \cup M, A_1 \sqcup A_2, g_1 \times g_2)$ where

$$(g_1 \times g_2)(\sigma, \omega) := (g_1(\sigma, \omega), g_2(\sigma, \omega)) \in \Sigma^{A_1 \cup A_2},$$

itself a solvable weakly neutralizable auxiliary circuit.
2) **Augmented constraints do not scale exponentially in A.** The traditional constraints (Equation 5) grow exponentially in A, but the constraint matrices of the g-augmented constraints 15) have no row-wise dependence on A and only grow quadratically columnwise.

Observation (1) means the function g can be built from libraries of known solvable weakly neutralizable functions, which shrinks the space of possible auxiliary functions so drastically that heuristic-driven brute force searches become computationally viable. Observation (2) means that the complexity of the linear solve grows only quadratically in A, and hence feasibility criterion are far cheaper to compute.

We call this the *gluing approach* to the reverse Ising problem.

The following two example auxiliary functions are practical in application.

Example 2.7. Suppose $g : \Sigma^N \times \Sigma^M \longrightarrow \Sigma^A$ is constant in the Σ^M component. Then any Ising system (X, R) which solves the circuit $(N \cup M, A, g)$ is trivially strongly neutralizable.

Example 2.8. Let $(\{a, b\}, \{c\}, AND)$ be the 1-bit AND circuit. There exists an Ising system $(\{a, b, c\}, R)$ which solves the circuit and is strongly neutralizable. Since the

substitution a for xy is a logical AND gate, Rosenberg reduction can therefore be viewed as the construction of solvable strongly neutralizable auxiliary functions through the successive gluing of AND gates.

3. Algorithms and Optimizations

Having established our approach to the reverse Ising problem, we now turn to the task of utilizing this theory to producing concrete solutions to specific circuits. Armed with Theorem 2.4, we will discuss implementations and improvements the basic approach described at the start of Section 2.3:

1) Make an initial guess $g : \Sigma^N \times \Sigma^M \longrightarrow \Sigma^A$ of neutralizable auxiliary function.
2) Measure the feasibility of g using a linear programming solver on the augmented constraints.
3) If g is feasible then we are done. Otherwise update the choice of g.

In Section 3.1 we discuss a modification of the linear problem which yields a feasibility heuristic useful for guiding the reassignment of g. Section 3.2 covers the main search algorithms used to construct a feasible g. Section 3.3 describes the design of a bespoke linear solver optimized for our specific problem. Sections 3.4 and 3.5 detail two further optimizations which offer significant improvements to the search algorithms.

3.1. Linear Problem with Artificial Variables

Recall the statement of our problem: we wish to find $g : \Sigma^{N+M} \longrightarrow \Sigma^A$ and a quadratic pseudo-Boolean polynomial $H : \Sigma^{N+M+A} \longrightarrow \mathbb{R}$ such that

$$H(\sigma, f(\sigma), g(\sigma, f(\sigma))) + 1 \leq H(\sigma, \omega, \eta) \quad (16)$$

for $\omega \neq f(\sigma)$. By the main theorem, this can be reduced to the set of constraints

$$H(\sigma, f(\sigma), g(\sigma, f(\sigma))) + 1 \leq H(\sigma, \omega, g(\sigma, \omega)) \quad (17)$$

as long as g is weakly neutralizable. Writing the Hamiltonian H as the inner product $H(\sigma) = \langle u, v(\sigma) \rangle$ as Equation 3, the constraints become

$$\langle u, \ v(\sigma, f(\sigma), g(\sigma, f(\sigma))) - v(\sigma, \omega, g(\sigma, \omega)) \rangle \leq -1 \quad (18)$$
$$\forall \sigma \in \Sigma^N, \omega \in \Sigma^M, \omega \neq f(\sigma) \quad (19)$$

Therefore, letting $B(f, g)$ be the constraint matrix whose rows are $v(\sigma, \omega, g(\sigma, \omega)) - v(\sigma, f(\sigma), g(\sigma, f(\sigma)))$, we wish to find a vector u such that $B(f, g)u \geq 1$. Since most choices of g will yield infeasible problems, we need a choice of *feasibility heuristic* to evaluate how close g is to generating a feasible problem. This can be done by adding new vector of variables ρ called the **artificial variables** to obtain the linear programming problem

$$\varrho(f, g) := \min_{\rho, u} \|\rho\|_1 \quad \text{s.t. } B(f, g)u + \rho \geq 1, \rho \geq 0 \quad (20)$$

The optimal value of the objective function is zero if and only if the choice of g is feasible. Otherwise, the optimal value of the objective function $\varrho(f, g)$ roughly measures how infeasible g is.

Remark 3.1. In Section 2.2.2, we discussed the fact that the linear problem for $M = 1$ is equivalent to fitting a linear hard-margin SVM. Adding the artificial variables to the $M = 1$ Ising circuit is in fact exactly equivalent to the linear soft-margin SVM.

3.2. Main Search Algorithms

We will use the notation $\times$ to denote the concatenation of tuples. Armed with the artificial variables heuristic, we can attempt to build a weakly neutralizable auxiliary map g that makes a given circuit (N, M, f) feasible. The simplest choice is a greedy algorithm based on threshold functions.

Algorithm 1 Greedy Search

Require: $\{\mathcal{T}_n\}_{n \in \mathbb{N}}$ sets of strongly neutralizable threshold functions on n variables.
Require: (N, M, f) Ising circuit.
$\quad g \leftarrow \varnothing$
$\quad$ **while** $\varrho(f, g) > 0$ **do**
$\quad\quad g \leftarrow g \times \arg\min_{a \in \mathcal{T}_{N+M+|g|}} \varrho(f, g \times a)$
$\quad$ **end while**

It follows from Section 2.2.1 that if

$$\mathcal{T}_n \supseteq \mathcal{A}_n := \{x \longmapsto x_i \wedge x_j | 1 \le i < j \le n\} \qquad (21)$$

then Algorithm 1 always terminates, since $\varrho(f, g) \ge \varrho(f, g \times a)$, for we can always set the coefficients on the new threshold function a to zero and recover the left hand side. It should be noted that requiring $\mathcal{T}_n$ grow with n is required for the algorithm to always terminate—fixing $\mathcal{T}_n = \mathcal{A}_{N+M}$ results in the algorithm never terminating when attempting to solve the parity check circuit with 3 input bits.

Remark 3.2. The $\mathcal{T}_n$ sets must be computed separately prior to the execution of Algorithm 1, see 3.2.1 for details. We use strongly neutralizable threshold functions for convenience; since the strong neutralizability property has no dependence on the underlying base circuit (N, M, f), and the $\mathcal{T}_n$ don't change from problem to problem. Using weakly neutralizable threshold functions should in principle yield a solution with fewer auxiliaries, since it gives many more options and thus increases the flexibility of the search. However, this adds significant overhead as the $\mathcal{T}_n$ sets must be computed for each new problem.

Note also that any auxiliary function g can be written componentwise as $(g_1, ..., g_A)$, where each $g_i : \Sigma^N \times \Sigma^M \longrightarrow \Sigma$ represents a single auxiliary bit. Since g is a solvable weakly/strongly neutralizable auxiliary function if its component functions are weakly/strongly neutralizable threshold functions, building g from its components always produces viable auxiliary circuits.

In practice, the greedy algorithm is clearly non-optimal. This is because ϱ is not additive with respect to its components: threshold functions have unpredictable synergies with each other and thus cannot be regarded as independent. That is, for threshold functions a and b, $\varrho(f, \varnothing) - \varrho(f, a \times b)$ differs unpredictably from $2\varrho(f, \varnothing) - \varrho(f, a) - \varrho(f, b)$, though the difference is usually not too large. Therefore, the greedy algorithm may result in earlier choices becoming non-optimal after later choices are made. This suggests a 'coordinate descent' type algorithm, Algorithm 2, which takes further advantage of the artificial variables heuristic to continually revise choices towards more optimal solutions. Additionally, it makes sense to expand the search space to weakly neutralizable functions, since gluing works in the exact same way.

Algorithm 2 Descent Algorithm

Require: (N, M, f) Ising circuit.
Require: $\mathcal{T}$ set of weakly neutralizable threshold functions with respect to circuit (N, M, f).
$$R \leftarrow (g, j, a) \longmapsto \left(\hat{g}_i = \begin{cases} g_i & \text{if } i \neq j \\ a & \text{if } i = j \end{cases} \right)_{1 \le i \le |g|}$$
$\quad g \leftarrow \varnothing$
$\quad S \leftarrow \{1, \dots, |g|\}$
$\quad$ **while** $\varrho(f, g) > 0$ **do**
$\quad\quad$ **if** $S = \varnothing$ **then**
$\quad\quad\quad g \leftarrow g \times \arg\min_{a \in \mathcal{T}} \varrho(f, g \times a)$
$\quad\quad\quad S \leftarrow \{1, \dots, |g|\}$
$\quad\quad$ **end if**
$\quad\quad j \leftarrow \arg\min_{i \in S} \varrho(f, g \setminus \{g_i\}) - \varrho(f, g)$
$\quad\quad \ell \leftarrow N + M + j - 1$
$\quad\quad \alpha \leftarrow \arg\min_{a \in \mathcal{T} \cup \mathcal{A}_\ell} \varrho(f, R(g, j, a))$
$\quad\quad$ **if** $\varrho(f, R(g, j, \alpha)) < \varrho(f, g)$ **then**
$\quad\quad\quad g_j \leftarrow \alpha$
$\quad\quad\quad S \leftarrow \{1, \dots, |g|\}$
$\quad\quad$ **else**
$\quad\quad\quad S \leftarrow S \setminus \{j\}$
$\quad\quad$ **end if**
$\quad$ **end while**

This is a much better algorithm in practice, though it will not always produce solutions with minimum $|g|$ due to getting stuck in local minima and therefore being forced to increase $|g|$. Due to the inclusion of the $\mathcal{A}_\ell$ sets, this algorithm will always terminate in finite time for the same reason as Algorithm 1. Plenty of tweaks can be made to the algorithm, most resulting in greater complexity:

- The heuristic values which are optimized over to set j are designed to select the auxiliary functions which are contributing the least—a more precise but far more computationally expensive option would be setting j by optimizing over the Shapley numbers of the g_i. Fast approximations of Shapley numbers do exist, but we leave experimentation with their implementation for future research.
- The symmetries discussed in Section 3.4 can be leveraged, in combination with a cache, to reduce the num-

ber of times that the expensive function ϱ is called (see Remark 3.4).

- The filtering method discussed in Section 3.5 can be implemented: We start with $\varrho = \varrho_i$ and increment i every time the condition $\varrho(f, g) = 0$ is satisfied, breaking out of the while loop only when $i = M$. This also significantly speeds up execution time.

3.2.1. Building Libraries of Threshold Functions. Remark 3.2 suggests that we precompute libraries of weakly neutralizable threshold functions for use in Algorithms 1 and 2. The problem of identifying all threshold functions of dimension d is still an open problem in general, but solved computationally for $d \leq 8$ [20]. For solving a specific problem, we can start with the known strongly neutralizable threshold functions such as AND and its self-dualization, then search through general threshold functions by testing the weak neutralizability condition using the linear programming solver. In practice, we observe that most threshold functions are not weakly neutralizable with respect to a given base circuit, and that the probability of compatibility decreases with the dimension of the threshold function. We can avoid redundancy in the search by exploiting the results of Section 3.4.

3.3. Linear Programming Solver

We found that no freely available linear programming solver was sufficiently fast or memory efficient to tackle the high volume of large-size linear problems needed to calculate the feasibility heuristic in our search algorithms. We wrote a multithreaded C implementation of the Mehrotra predictor-corrector based on [19] specifically optimized for our problem. It is 2-3 times faster than GLOP [22] for highly overdetermined sparse sign matrices like the Ising constraint sets, and significantly more memory efficient. Details of the solver design can be found in the eprint version of this paper [15, Section 6.2]. It is inspired by the work in [23], but trades away memory optimization for greater flexibility in choosing the coefficient matrix, among several other modifications designed to suit our problem in particular.

3.4. Symmetries of Ising Circuits

We set $\Sigma = \{\pm 1\}$ unless otherwise noted. Fix a circuit (N, M, f). Notice that an auxiliary function $g : \Sigma^N \times \Sigma^M \longrightarrow \Sigma^A$ is solvable and weakly neutralizable with Hamiltonian H if and only if $-g$ is solvable and weakly neutralizable with Hamiltonian H' given by flipping the signs of every coefficient of an auxiliary local bias, auxiliary/input interaction or auxiliary/output interaction. Hence if g is determined to be infeasible then $-g$ must also be infeasible without performing any additional computation. This is an example of an Ising symmetry: a object α acting on both circuits and Ising systems such that such that (N, M, f) is solvable by H if and only if $(N, M, \alpha f)$ is solvable by $(X, \alpha H)$. We discuss two such Ising symmetries and

their use in further reducing the problem complexity. Proofs and further mathematical details can be found in the eprint version of this article [15, Section 6.1].

3.4.1. Spin Actions. Viewing Σ^X as the set of functions $X \longrightarrow \Sigma$, for each $\alpha \in \Sigma^X$ we obtain an action on Σ^X via pointwise multiplication by α:

$$\alpha\sigma(x) = \alpha(x) \cdot \sigma(x). \tag{22}$$

This is called a *spin action*, and if spin states are instead viewed as tuples of ± 1 then this is nothing more than the component-wise multiplication of $\{\pm 1\}^n$. Decomposing α into input and output components gives an action on f by

$$(\alpha f)(\sigma) := \alpha|_M \cdot f(\alpha|_N \sigma). \tag{23}$$

We get a corresponding action on Ising systems (X, H) by multiplying $v(\alpha)$ componentwise with u, the coefficient vector of H:

$$\alpha u := v(\alpha) \cdot p. \tag{24}$$

Thus defined, α is an Ising circuit symmetry. The group of all spin actions on Σ^X is easily seen to be isomorphic to $(\mathbb{Z}/2\mathbb{Z})^n$, where $|X| = n$

3.4.2. Coordinate Permutations. Given a permutation α of N and a permutation β of M, we can define an action on $f : \Sigma^N \longrightarrow \Sigma^M$ by

$$(\beta f \alpha)(\sigma) = f(\sigma \circ \alpha) \circ \beta. \tag{25}$$

The corresponding action on the coefficient vector p of H is easier to write using the h and J convention (see Equation 2). We define the action of α on h and J to be the identity action, and we define β to act by index permutation

$$\beta h_i := h_{\beta(i)}, \quad \beta J_{i,j} := J_{\beta(i), \beta(j)} \tag{26}$$

setting $\beta(i) = i$ whenever $i \in N$. This makes coordinate permutations Ising circuit symmetries. The group of all coordinate permutations is the product of permutation group of orders N and M: $S_N \times S_M$.

3.4.3. Symmetry Speedups. Denote by $\mathcal{G}(N, M)$ the group of *Ising symmetries* obtained by compositions of spin actions and coordinate permutations for circuits of the form (N, M, f) and by $\mathcal{S}(N, M)$ the group of spin actions. For proof, we again direct the reader to [15]. The following proposition and remark illustrate how symmetries of Ising circuits can speed up our algorithms.

Proposition 3.3. If $\alpha \in \mathcal{G}(N, M)$ then $g : \Sigma^N \longrightarrow \Sigma$ is a weakly/strongly neutralizable threshold function if and only if αg is as well. Additionally, if $\alpha \in \mathcal{S}(N \cup M, A)$ is a spin action on the auxiliary circuit $(N \cup M, A, g)$, then $\varrho(f, g) = \varrho(f, \alpha g)$.

Remark 3.4. This proposition speeds up the search for auxiliary functions in two ways.

1) **Building libraries of threshold functions.** When computing sets $\mathcal{T}$ of threshold functions in Algorithms 1 and 2, one need only test one candidate threshold function g for solvability and neutralizability to determine the solvability and neutralizability of the entire orbit. This eliminates

$$|\mathcal{G}(N, M)| = 2^{N \cup M} \cdot N! \cdot M! \tag{27}$$

linear solves in the best case scenario, but far fewer in practice since the action of $\mathcal{G}(N, M)$ is not free.

2) **Computing the feasibility heuristic.** Because spin actions preserve ϱ, two auxiliary functions g and g' related by a spin action are *indistinguishable by the feasibility heuristic*. One can therefore take $\mathcal{T}$ to contain only one weakly/strongly neutralizable threshold function in each $\mathcal{S}(N, M)$ orbit without losing any performance. This eliminates up to $|\mathcal{S}(N, M)| = 2^{N+M} - 1$ computations of ϱ.

3.5. Constraint Filtering

The linear problem of checking the feasibility of an augmented constraint set for circuit (N, M, f) with A auxiliaries has difficulty $\mathcal{O}(2^{N+M}(N + M + A)^2)$. If we need to search through a large amount of auxiliary functions, this may still be quite expensive. In practice, however, it turns out that the artificial variables are not evenly distributed across the rows of the constraint matrix, and therefore we can cut down on the factor of 2^M quite substantially.

For a fixed circuit (N, M, f) and an infeasible auxiliary function $g : \Sigma^{N+M} \longrightarrow \Sigma^A$, the constraint matrix B serves only to verify the infeasibility of g and compute the criterion ϱ. This matrix is quite tall: B has $2^N \cdot (2^M - 1)$ rows but only $M + A + \binom{N+M+A}{2} - \binom{N}{2}$ columns. This results in an exceptionally over-determined system in which we expect many redundant constraints. Now consider a matrix B' consisting of only a subset of the rows in B and the reduced problem $B'u \geq 1$ with feasibility criterion ϱ'. Evidently ϱ' is a lower bound on ϱ which is cheaper to compute than ϱ. Therefore we may reasonably want to know: *(i)* How well does ϱ' approximate ϱ, as a function of the number of constraints removed from B—and relatedly, *(ii)* If $\varrho' = 0$, what is the probability that $\varrho = 0$? If it is in fact a good approximation, we can use it as an approximate criterion for the first phase of the algorithm and thus save a lot of time. This is indeed the case: Figure 1 demonstrates that smaller constraint sets serve as good approximations of 1 and Figure 2 illustrates that they detect infeasibility with high probability.

As the size of A is increased (i.e. as threshold functions are added in Algorithms 1 and 2 and g becomes more feasible) we expect that the requisite size of the matrices in this filtration will also grow. It is therefore useful to consider a series of smaller linear problems of various sizes obtained by incrementally adding constraints from B; that is, to consider a filtration $\{B_i\}_{i \in [0..k]}$ of the constraint matrix B rather than a single reduced matrix B'.

Each row of B is defined by a choice of input $\sigma \in \Sigma^N$ and a choice of incorrect output $\omega \in \Sigma^M, \omega \neq f(\sigma)$. In practice, rows whose incorrect output ω is closer in Hamming distance to the correct output $f(\sigma)$ contribute more to the heuristic ϱ and are collectively more difficult to satisfy. Therefore, a natural choice of filtration is $\{B_i\}_{1 \leq i \leq M}$, where B_i is the matrix consisting of all rows defined by choices of incorrect output ω such that $d(\omega, f(\sigma)) \leq i$ and d denotes Hamming distance. We call B_i the 'ith local constraints' because it requires that $(\sigma, f(\sigma))$ be the absolute minimum of the input level $\mathcal{L}_\sigma$ on the Hamming ball of radius i around σ, for every σ. Note that B_1 has M rows per input level, and B_i adds $\binom{M}{i}$ rows to each input level relative to B_{i-1}.

It turns out that the artificial variables are mostly concentrated in the local constraints with i small (see Figure 1), which also have far fewer rows. Therefore, testing with the first or second local constraints provides a reasonably good lower bound to the total sum of artificial variables for the whole problem at a fraction of the computational cost. If we filter possible candidate auxiliary function candidates with the X_1 constraint set, each linear program solve takes has difficulty $\mathcal{O}(2^N M(N + M + A)^2)$, an improvement over the whole problem by a factor of $2^M/M$. In practice, we first construct a solution which satisfies B_1 or B_2 which we then use as a starting point to search for a solution to B, thus saving a significant amount of effort.
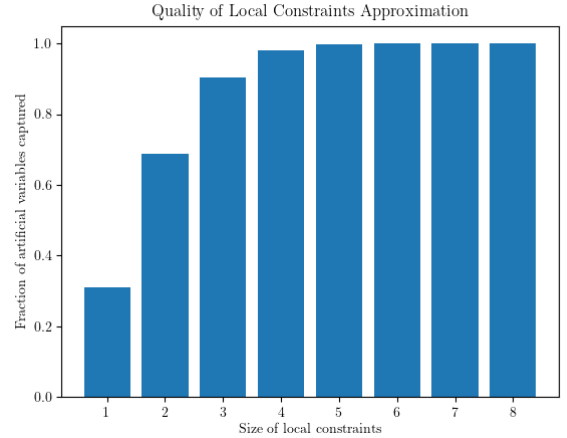

Figure 1. Proportion of artificial variables captured by the ith local constraints, ϱ_i/ϱ, plotted against i for the problem of 4×4 integer multiplication with 12 auxiliaries, sampled randomly from a set of weakly neutralizable threshold functions, averaged over 50 runs. Observe that $i = 3$ captures almost all of the artificial variables.

4. Results

The implementation of integer multiplication on Ising hardware turns out to be quite difficult, and hence serves as a good benchmark for our general solution methods. We denote the problem of finding sets of a threshold functions which solve the problem of $n \times m$ integer multiplication
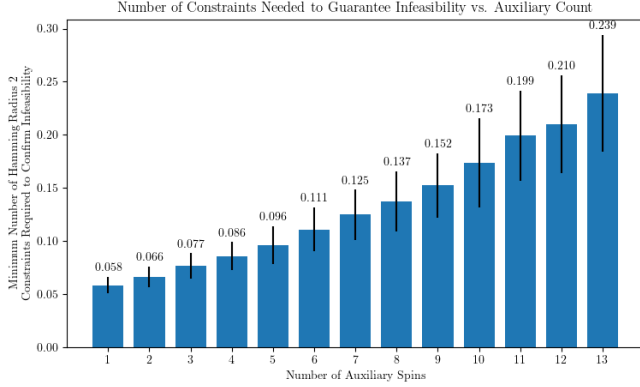
Figure 2. The minimum number of hamming radius 2 constraints required to confirm the infeasibility of randomly chosen auxiliary functions of specified sizes, averaged across 500 runs. The minimum number of constraints needed to detect infeasibility was obtained by performing a binary search on a random maximal filtration of B_2, the 2nd local constraints. We emphasize that the y-axis measures the number of requisite constraints as a fraction of all 2nd local constraints; the collection of all 2nd local constraints themselves only account for $\frac{9216}{65280} \approx 0.1412$ of the total number of constraints.

as MUL$[n \times m \times a]$. Our methods produced many solutions to MUL$[3 \times 3 \times 3]$, MUL$[3 \times 4 \times 5]$, and MUL$[4 \times 4 \times 12]$. In each case, our solution represents the current optimum in total spin count by a large margin. Previously, Andriyash [1] constructed solutions to MUL$[3 \times 3 \times 42]$ and MUL$[4 \times 4 \times 88]$ on a D-Wave system. Our results represent a significant reduction in total circuit size, from 54 to 15 total spins for 3×3 multiplication and from 104 to 28 total spins for 4×4. Therefore, our results reduce the total circuit layout area by a factor of around 3.5 compared to previous designs. By reducing the total number of spins substantially, we have decreased the cost of implementing such circuit in hardware.

4.1. Runtimes & Example Data

Experiments were run on dual socket compute nodes with 64 cores per AMD EPYC 7713 socket and two threads per core for a total of 256 processors per node. Average runtimes for successful solutions to our benchmark problems were as follows:

Problem	Average Runtime (s)
MUL$[3 \times 3 \times 3]$	902
MUL$[3 \times 4 \times 5]$	23,520
MUL$[4 \times 4 \times 12]$	67,890

Figure 3 graphically shows example solutions to the three benchmark problems. Each row depicts a single threshold function's weight vector, showing how much that threshold function depends on each of the original $N + M$ spins.

5. Conclusion

Our main theoretical result has made computationally tractable the large-scale searches required to solve nontrivial cases of the Reverse Ising problem. We can now algorithmically discover far more compact Ising Hamiltonians
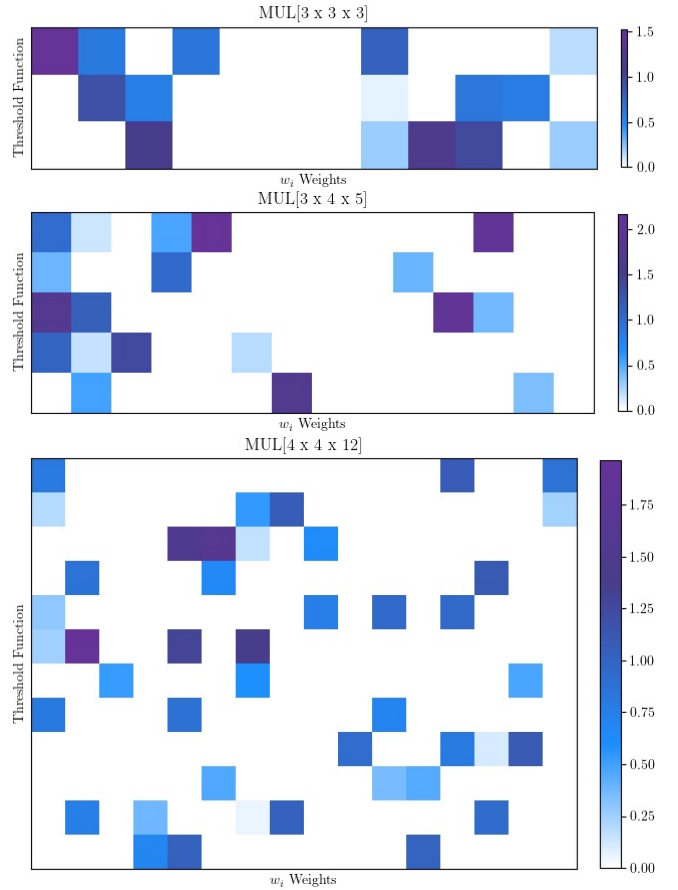

Figure 3. Examples of feasible sets of sparse auxiliary threshold functions for each of our benchmark problems. Each row is the weight vector w of a threshold function.

which realize the desired circuity than would be possible to construct by hand. This allowed us to significantly shrink the best known minimal-spin solutions to the Ising formulations of integer multiplication circuits, improving on previously known results by a factor more than a factor of 3. However, we believe doing better is possible. Relaxing to more general classes of auxiliary functions beyond tuples of threshold functions will likely allow for even smaller circuits, though it is more mathematically difficult. We also wish to find ways to incorporate practical considerations such as energy gap, dynamic range, and graph structure into our formulation.

The practical feasibility of Ising computing, like other probabilistic, distributed, and quantum-inspired approaches, depends on the development of a strong theoretical foundation for the analysis of circuit design problems. We believe that this field is still in its infancy, that the mathematics is far from settled, and especially that we have only captured a fraction of the the potential power of auxiliary maps for compactly realizing general functions. It is our hope that this paper will help to contribute both to the development of this theory and to the search for practical methods for programming the computers of the future.

Acknowledgments

The authors would like to thank Dr. Karin Rabe and Dr. Gregory Moore (Department of Physics, Rutgers University) and Luisa Velasco (Department of Mathematics, University of Texas at Austin) for their feedback and editorial help.

References

[1] E. Andriyash, Z. Bian, F. A. Chudak, A. D. King, and W. G. Macready. Boosting integer factoring performance via quantum annealing o sets technical report. 2016. URL https://api.semanticscholar.org/CorpusID:44099134.

[2] P. Baldi and R. Vershynin. Polynomial threshold functions, hyperplane arrangements, and random tensors. *SIAM Journal on Mathematics of Data Science*, 1(4):699–729, 2019. doi: 10.1137/19M1257792. URL https://doi.org/10.1137/19M1257792.

[3] M. K. Bashar and N. Shukla. Designing Ising machines with higher order spin interactions and their application in solving combinatorial optimization. *Scientific Reports*, 13(9558), 2023. doi: 10.1038/s41598-023-36531-4.

[4] J. D. Biamonte. Nonperturbative k-body to two-body commuting conversion hamiltonians and embedding problem instances into Ising spins. *Phys. Rev. A*, 77:052331, May 2008. doi: 10.1103/PhysRevA.77.052331.

[5] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, and A. Roy. Discrete optimization using quantum annealing on sparse Ising models. *Frontiers in Physics*, 2, 2014. ISSN 2296-424X. doi: 10.3389/fphy.2014.00056. URL https://www.frontiersin.org/articles/10.3389/fphy.2014.00056.

[6] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1):155–225, 2002. ISSN 0166-218X. doi: 10.1016/S0166-218X(01)00341-9.

[7] E. Boros, Y. Crama, and E. Rodríguez-Heck. Compact quadratizations for pseudo-boolean functions. *Journal of Combinatorial Optimization*, 39(3):687–707, 2020. doi: 10.1007/s10878-019-00511-0. URL https://doi.org/10.1007/s10878-019-00511-0.

[8] S. G. Brush. History of the lenz-Ising model. *Rev. Mod. Phys.*, 39:883–893, Oct 1967. doi: 10.1103/RevModPhys.39.883. URL https://link.aps.org/doi/10.1103/RevModPhys.39.883.

[9] B. Cai, Y. He, and Y. X. et. al. Unconventional computing based on magnetic tunnel junction. *Applied Physics A*, 129(236), mar 2023. doi: 10.1007/s00339-022-06365-4.

[10] J. Chou, S. Bramhavar, S. Ghosh, and W. Herzog. Analog coupled oscillator based weighted Ising machine, 2019.

[11] N. Dattani. Quadratization in discrete optimization and quantum mechanics, 2019.

[12] N. Gruzling. Linear separability of the vertices of an n-dimensional hypercube, 2007.

[13] M. Gu and Á. Perales. Encoding universal computation in the ground states of Ising lattices. *Physical Review E*, 86(1), jul 2012. doi: 10.1103/physreve.86.011116.

[14] A. Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2, 2014. ISSN 2296-424X. doi: 10.3389/fphy.2014.00005.

[15] I. K. Martin, A. G. Moore, J. T. Daly, J. J. Meyer, and T. M. Ranadive. Design of general purpose minimal-auxiliary ising machines. 2023. URL https://arxiv.org/abs/2310.16246.

[16] P. L. McMahon, A. Marandi, Y. Haribara, R. Hamerly, C. Langrock, S. Tamate, T. Inagaki, H. Takesue, S. Utsunomiya, K. Aihara, R. L. Byer, M. M. Fejer, H. Mabuchi, and Y. Yamamoto. A fully programmable 100-spin coherent Ising machine with all-to-all connections. *Science*, 354(6312):614–617, 2016. doi: 10.1126/science.aah5178.

[17] C. Monroe, W. Campbell, L.-M. Duan, Z.-X. Gong, A. Gorshkov, P. Hess, R. Islam, K. Kim, N. Linke, G. Pagano, P. Richerme, C. Senko, and N. Yao. Programmable quantum simulations of spin systems with trapped ions. *Reviews of Modern Physics*, 93(2), apr 2021. doi: 10.1103/revmodphys.93.025001. URL https://doi.org/10.1103%2Frevmodphys.93.025001.

[18] L. M. N. Kodama. Scaling expander networks. *In preparation for submission to the 40th International Conerference on Machine Learning (ICML 2024)*.

[19] J. Nocedal and S. J. Wright. *Linear Programming: Interior-Point Methods*, pages 392–420. Springer New York, New York, NY, 2006. ISBN 978-0-387-40065-5. doi: 10.1007/978-0-387-40065-5_14.

[20] OEIS Foundation Inc. Number of threshold functions of n or fewer variables, Entry A000609 in The On-Line Encyclopedia of Integer Sequences, 2023. URL https://oeis.org/A000609.

[21] S. Patel, L. Chen, P. Canoza, and S. Salahuddin. Ising model optimization problems on a fpga accelerated restricted boltzmann machine, 2020.

[22] L. Perron and V. Furnon. Or-tools, 08 2023. URL https://developers.google.com/optimization/.

[23] T. M. Ranadive, J. T. Daly, J. J. Meyer, A. G. Moore, and I. K. Martin. Enabling Ising machine arithmetic with high performance computing. *unpublished*, 2023.

[24] I. G. Rosenberg. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'Etudes de Recherche Operationnelle*, 17(71-74), 1975.

[25] T. Wang and J. Roychowdhury. Oim: Oscillator-based Ising machines for solving combinatorial optimisation problems, 2019.

[26] J. D. Whitfield, M. Faccin, and J. D. Biamonte. Ground-state spin logic. *Europhysics Letters*, 99(5), sep 2012. doi: 10.1209/0295-5075/99/57004.