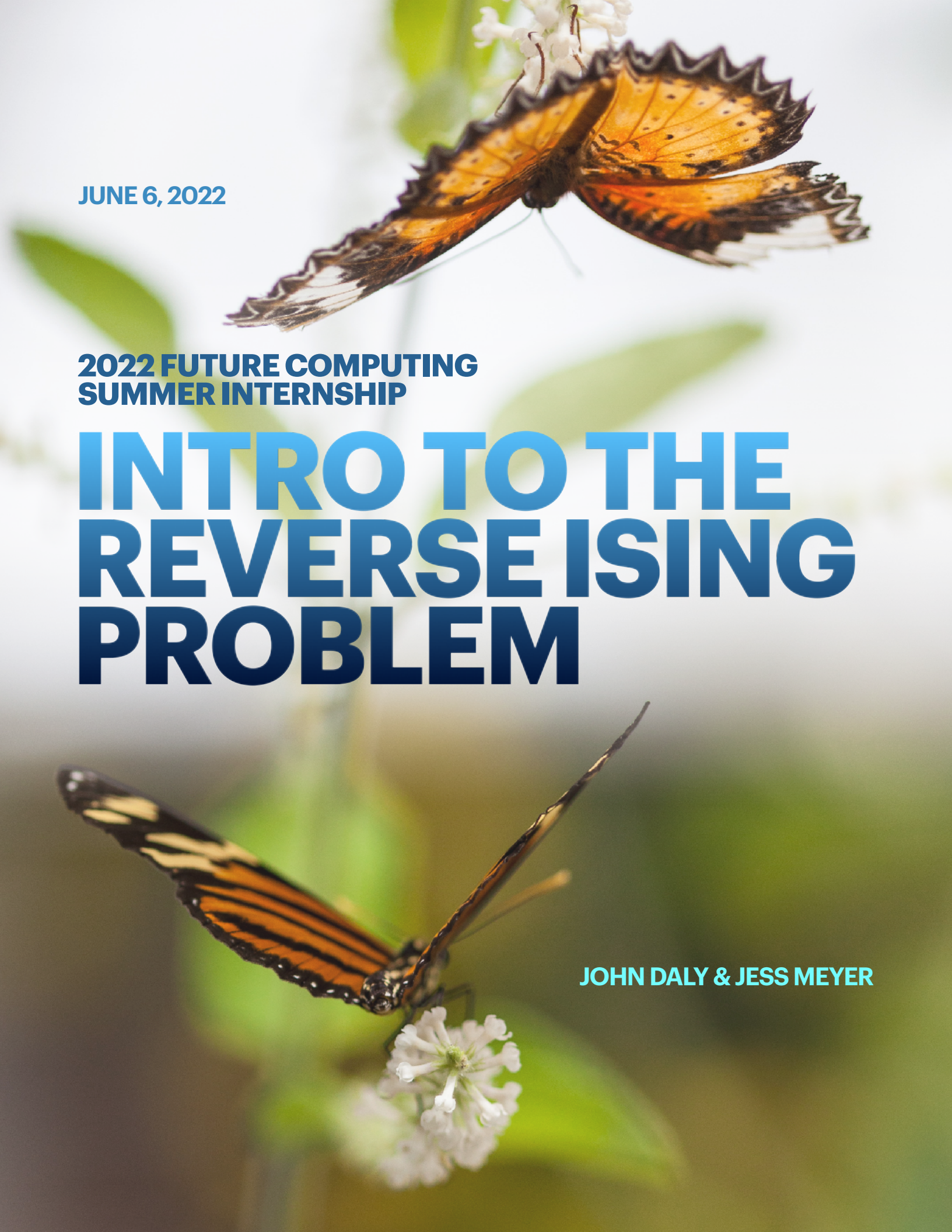


JUNE 6, 2022

2022 FUTURE COMPUTING SUMMER INTERNSHIP

INTRO TO THE REVERSE ISING PROBLEM

JOHN DALY & JESS MEYER

TABLE OF CONTENTS

The Ising Machine Learning Project

2022 FUTURE COMPUTING SUMMER INTERNSHIP

The Probabilistic Computing team at the Laboratory for Physical Sciences (LPS) has spent the last several years exploring solutions to the reverse Ising problem. We currently have a solution technique using an internally developed two-stage algorithm that first searches for a set of feasible parameters and then solves a system of constraints derived from the feasible parameters. Both stages have exponential complexity, but our team found ways to improve the solution time from 120 days to less than an hour by reducing the problem to polynomial complexity. We also successfully solved a system of 267 million constraints in a total solve time of 27 days using 5.5 terabytes of shared memory. Our goal for this summer: Use machine learning to do even better.

THE REVERSE ISING PROBLEM

The Ising model is a mathematical model of ferromagnetism in statical mechanics. It was invented by Wilhelm Lenz in 1920 and first solved by his PhD student Ernst Ising in 1924. One example of such a system is the interaction between groups of magnetic spins in an electromagnetic field, as depicted in Figure 1. An Ising computer is a physical system with controllable inputs and outputs that can be used to perform computation. Because such systems can transition to new states at extremely low energies, as low as 6 zJoule ($6x10^{-21}$ Joules) at room temperature, Ising computation provides a mechanism to perform highly energy efficient computation by utilizing a non-vonNeumann model of computation.[1]

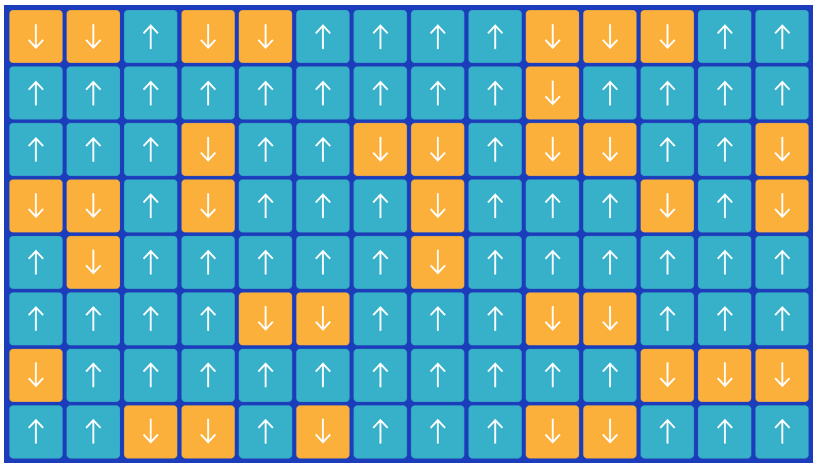


FIGURE 1. An Ising model is a lattice of spins, some spin-up, or taking the value 1, and some spin-down, or taking the value -1.

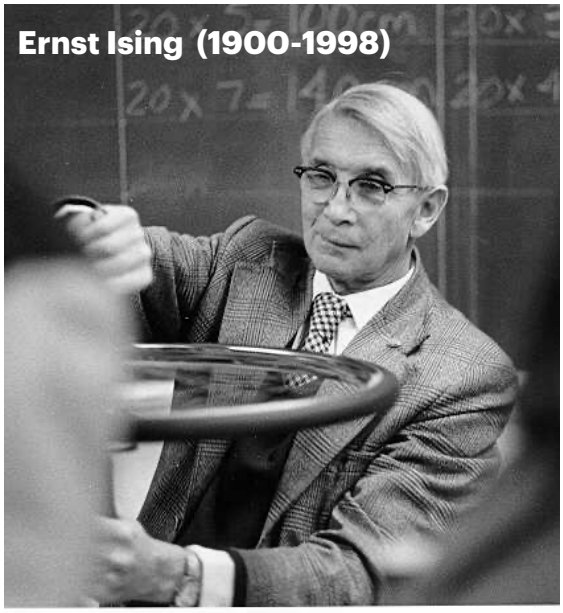
INTRODUCING THE ISING MODEL

An Ising system will naturally evolve to its lowest possible energy configuration. For each fixed set of input spins, the remaining spins will evolve to a configuration of up and down spins. The probability of any particular configuration occurring depends on the the relative energies of all possible configurations, which in turn depends on the local biases and coupling strengths. In order to design Ising-based logic circuits one needs to be able

[1] McGoldrick, Brooke C. et al. "Ising Machine Based on Electrically Coupled Spin Hall Nano-Oscillators." Physical Review Applied (2022).

to describe a system with a fixed set of inputs for which the desired output is the minimum energy spin configuration of all the possible state configurations. The energy is given by the Hamiltonian as follows.

$$H = \sum_{i=1}^{n} h_i s_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} s_i J_{i,j} s_j \qquad s_i \in \{-1,1\} \quad h_i \in \mathbb{R} \quad J_{i,j} \in \mathbb{R}$$



Consider the Ising model describing the total energy of the system as a function of n spins s_i, which can have one of two possible values—call them $+1$ for *up* or -1 for *down*, and the *local biases* h_i and *coupling strengths* $J_{i,j}$ of the magnets, which can be positive or negative. When one uses the Ising model with a fixed set of biases and coupling strengths to compute the values of the spins, this is known as the *forward Ising problem*. This is the formulation of the Ising problem that is focused on in statistical mechanics. The goal is to study the expected equilibrium configuration; i.e., the configurations that the system is most likely to stabilize at. This is a quadratic integer programming problem.

In the *reverse Ising problem*, the goal is to determine the local biases and coupling strengths that would cause the system to have these desired output states as the equilibrium states. When feasible, this is a linear problem. However, as stated the problem is typically not feasible. In order to make the problem solvable, additional degrees of freedom are required. These are obtained by adding additional spins called *auxiliary spins* to the system that have no role in the operation being performed other than to increase the number of degrees of freedom. For every auxiliary spin added to the system, the number of configurations that correspond to each correct solution double.

The values of the auxiliary spins is irrelevant to the computation being performed. However, the Hamiltonian is now a function of the states of the auxiliary spins. Thus, in the process of calculating the local biases and coupling strengths, it is necessary to

compute feasible sets of states of the auxiliary spins. Consequently, it is necessary to solve for the local biases, coupling strengths, and the states of the auxiliary spins simultaneously. This converts the problem from a linear problem into one that is mixed integer (since the auxiliaries are binary variables) and either a quadratic or a cubic programming problem.

A SIMPLE EXAMPLE: DESIGN OF A LOGICAL AND CIRCUIT

Starting with a simple example, consider how one might design a logical AND gate using Ising logic. Begin by examining the truth table associated with AND given as following.

$$0 \wedge 0 = 0 \qquad 0 \wedge 1 = 0 \qquad 1 \wedge 0 = 0 \qquad 1 \wedge 1 = 1$$

The spin interactions of our Ising system will not be limited to nearest neighbor. Spin interactions will instead be described by an all-to-all connected graph with edges that are the coupling strengths and vertices that are local biases. Further, assume the biases of and coupling strengths of the input spins are always zero, as illustrated in Figure 2.

AND Truth Table

s_1	s_2	s_3
-1	-1	-1
-1	$+1$	-1
$+1$	-1	-1
$+1$	$+1$	$+1$

Our goal is to configure the local and interaction potentials in such a way that the orientation of the blue spin in at its ground state will correspond to the logical AND of the two input spins. Begin by translating the truth table into spin configurations for the Ising system. By choosing spin down to be 0 or False, and spin up to be 1 or True, the truth table may be rewritten in terms of Ising spin configurations, where s_1 and s_2 are inputs while s_3 is the output.

The next step is to enforce the constraint that the spin configurations corresponding to the minimum energy configurations of the Ising system are the same as the spin states found in the truth table for our logic circuit. This is equivalent to requiring that for each given input spin configuration, the configuration of the desired output configuration has lower energy than all other possible output configurations. To formalize this we require that the Hamiltonian of every incorrect solution corresponding to a given set of input spins is at least one less than the Hamiltonian of the correct solutions corresponding to those input spins.

$$H(-1,-1,+1) - H(-1,-1,-1) \geq 1$$
$$H(-1,+1,+1) - H(-1,+1,-1) \geq 1$$
$$H(+1,-1,+1) - H(+1,-1,-1) \geq 1$$
$$H(+1,+1,-1) - H(+1,+1,+1) \geq 1$$

In theory, the energy gap between desired and undesired energy configurations could be arbitrarily close to zero. In practice, solving numerically for systems constrained by zero can be quite difficult so the difference between the Hamiltonians of the incorrect and correct states is arbitrarily bounded by one, without loss of generality. Note that since the problem does not correspond to a physical system at the moment the energy values are currently non-dimensional. Alternatively, this constraint satisfaction problem written as a linear system of inequalities as follows.

$$\begin{bmatrix} 1 & -1 & -1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} h_3 \\ J_{1,3} \\ J_{2,3} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

This system has infinitely many solutions, but any feasible solution is sufficient for our purposes. One such solution is given as follows. Configuring the system in Figure 2 with these values for local biases and coupling strength, the spin of the blue output value will always equal the logical AND of the spins of the two red input values.

$$\begin{bmatrix} h_3 \\ J_{1,3} \\ J_{2,3} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

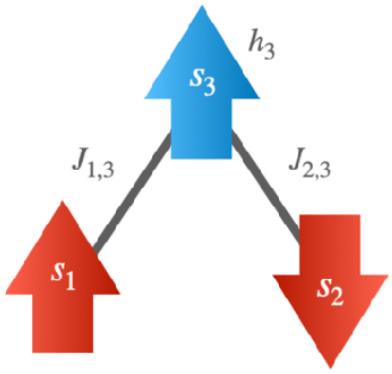


FIGURE 2. A 1-bit logical AND gate consisting of two input spins (red) and an output spin (blue).

A MORE COMPLICATED EXAMPLE: DESIGN OF A LOGICAL XOR CIRCUIT

For this next example, apply the approach taken for logical AND applying it to logical XOR. The new truth table becomes the following.

$$0 \oplus 0 = 0 \qquad 0 \oplus 1 = 1 \qquad 1 \oplus 0 = 1 \qquad 1 \oplus 1 = 0$$

Enforce the constraint that the spins corresponding to the least energy configurations of our Ising system are the same as the spin states found in the truth table as in the previous example. s_1 and s_2 are input spins. s_3 is the output spin. This is done as follows.

XOR Truth Table

s_1	s_2	s_3	a_1
-1	-1	-1	± 1
-1	$+1$	$+1$	± 1
$+1$	-1	$+1$	± 1
$+1$	$+1$	-1	± 1

$$\begin{aligned}
H(-1,-1,+1) - H(-1,-1,-1) &\geq 1 \\
H(-1,+1,-1) - H(-1,+1,+1) &\geq 1 \\
H(+1,-1,-1) - H(+1,-1,+1) &\geq 1 \\
H(+1,+1,+1) - H(+1,+1,-1) &\geq 1
\end{aligned}$$

This system of equations can be written as a linear system of inequalities as follows.

$$\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} h_3 \\ J_{1,3} \\ J_{2,3} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Unlike the system for logical AND, **the system above has no feasible solutions**. In order to make feasible solutions possible we need to create more degrees of freedom in the system. Do this by adding an auxiliary spin as illustrated in Figure 3. The truth table for logical XOR needs to be translated into spin configurations for the Ising system keeping in mind that a configuration consists of values for input, output, and auxiliary spins which are allowed to assume any value. Choosing spin down to be 0 or False, and spin up to be 1 or True, the truth table can be rewritten as follows, where s_1 and s_2 are input spins, s_3 is the output spin, and a_1 is the newly introduced auxiliary spin.

To specify the energy constraints for this new system with the auxiliary spin, **the energy of at least one correct configuration for each combination of input spins must be less than the energy of all the incorrect configurations corresponding to those input spins**. There are $2^1 = 2$ auxiliary states and $2^2 = 4$ input states. So the number of constraints will be $2 \times 4 = 8$. The single variable a_1 has been replaced by an array of variables $\alpha_{1,j}$ because the configuration of the auxiliary spins is allowed to change from input to input.

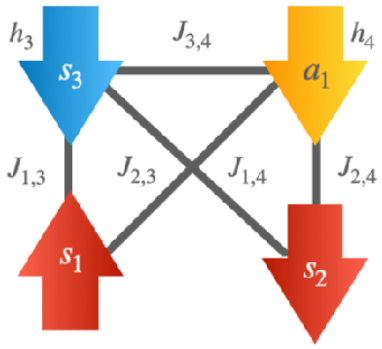


FIGURE 3. A 1-bit logical XOR gate consisting of two input spins (red) an output spin (blue) and an auxiliary spin (yellow).

$$
\begin{aligned}
&H(-1,-1,+1,+\alpha_{1,1}) - H(-1,-1,-1,+\alpha_{1,1}) \geq 1\\
&H(-1,-1,+1,-\alpha_{1,1}) - H(-1,-1,-1,+\alpha_{1,1}) \geq 1\\
&H(-1,+1,+1,+\alpha_{1,2}) - H(-1,+1,-1,+\alpha_{1,2}) \geq 1\\
&H(-1,+1,+1,-\alpha_{1,2}) - H(-1,+1,-1,+\alpha_{1,2}) \geq 1\\
&H(+1,-1,+1,+\alpha_{1,3}) - H(+1,-1,-1,+\alpha_{1,3}) \geq 1\\
&H(+1,-1,+1,-\alpha_{1,3}) - H(+1,-1,-1,+\alpha_{1,3}) \geq 1\\
&H(+1,+1,-1,+\alpha_{1,4}) - H(+1,+1,+1,+\alpha_{1,4}) \geq 1\\
&H(+1,+1,-1,-\alpha_{1,4}) - H(+1,+1,+1,+\alpha_{1,4}) \geq 1
\end{aligned}
$$

This constraint satisfaction problem can be expressed as a mixed integer quadratic system of inequalities as follows.

$$
\begin{bmatrix}
1 & 0 & -1 & 0 & -1 & 0 & \alpha_{1,1}\\
1 & -\alpha_{1,1} & -1 & \alpha_{1,1} & -1 & \alpha_{1,1} & 0\\
-1 & 0 & 1 & 0 & -1 & 0 & -\alpha_{1,2}\\
-1 & -\alpha_{1,2} & 1 & \alpha_{1,2} & -1 & -\alpha_{1,2} & 0\\
-1 & 0 & -1 & 0 & 1 & 0 & -\alpha_{1,3}\\
-1 & -\alpha_{1,3} & -1 & -\alpha_{1,3} & 1 & \alpha_{1,3} & 0\\
1 & 0 & 1 & 0 & 1 & 0 & \alpha_{1,4}\\
1 & -\alpha_{1,4} & 1 & -\alpha_{1,4} & 1 & -\alpha_{1,4} & 0
\end{bmatrix}
\cdot
\begin{bmatrix}
h_3\\ h_4\\ J_{1,3}\\ J_{1,4}\\ J_{2,3}\\ J_{2,4}\\ J_{3,4}
\end{bmatrix}
\geq
\begin{bmatrix}
1\\1\\1\\1\\1\\1\\1\\1
\end{bmatrix}
$$

The quadratic system of constraints above is provably non-convex, though the proof is beyond the scope of this introduction[2]. The consequence of this non-convex property is that between any two feasible points that satisfy the system of equations the intervening points in general will not be feasible. This in turn means that there will be no general polynomial time algorithm for solving the system, because the ability to find one solution gives no guarantees or information about where to look for other solutions.

The methods adopted by our research team for solving this system all involve using heuristic approaches that trades optimality and completeness for convenience and speed. We first identify feasible solutions for the values of $\alpha_{i,j}$ and then solve the local

[2] A convex system of quadratic constraints must have a Hessian matrix which is positive semi-definite.

biases h_i and coupling strengths $J_{i,j}$ in the resulting linear system. Using this method leads to feasible solution such as the following.

$$\begin{bmatrix} \alpha_{1,1} \\ \alpha_{1,2} \\ \alpha_{1,3} \\ \alpha_{1,4} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} h_3 \\ h_4 \\ J_{1,3} \\ J_{1,4} \\ J_{2,3} \\ J_{2,4} \\ J_{3,4} \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \\ -1 \\ -2 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

For this system there are $2^4 = 16$ possible auxiliary combinations for $\alpha_{i,j}$ but only half of those are feasible combinations that lead to a solution. There is no known method of distinguishing feasible from infeasible auxiliary combinations for an arbitrary mixed integer non-linear system at present, which is one of the significant motivations for the Ising machine learning project.

CHARACTERIZING SCALING IN TERMS OF PROBLEM INPUT PARAMETERS

Reverse Ising problems are named using the convention `<op>_<i>x<j>x<k>` where `<op>` is one of `{AND, OR, XOR, ADD, MUL}` and `<i>`, `<j>`, and `<k>` are respectively the number of input spins for the first operand (i), input spins for the second operand (j), and auxiliary spins (k). The number of input spins (m), output spins (n), and the total number of spins (l) are then given as follows. Notice that this assumes neither the multiplicand or multiplier is 1. If either is 1 then logical MUL is equivalent to logical AND.

$$m = i + j \qquad n = \begin{cases} \max(i,j) + 1 & \text{for <op> = ADD} \\ i + j & \text{for <op> = MUL} \\ \max(i,j) & \text{for all other <op>} \end{cases} \qquad l = m + n + k$$

The number of local biases (h_i), excluding inputs that are zero, is $l - m$. The number of coupling strengths ($J_{i,j}$), excluding those between inputs which do not interact, is given

by $\frac{1}{2}(n+k)(m+l-1)$. Together, these are referred to as the *potential variables*. The total number of auxiliary variables (K) is the number of auxiliary spins times the number of input configurations. The total number of potential variables (M) is the sum of the number of local bias variables and the number of coupling strength variables. The total number of inequality constraints (N) is the number of input configuration times the number of auxiliary configurations times the number of incorrect output configurations. These are summarized as follows.

$$K = k2^m \quad M = \frac{1}{2}(n+k)(m+l+1) \quad N = 2^{m+k}(2^n-1)$$

Now the constraint satisfaction problem can be generalized in terms of an $N \times M$ auxiliary constraint matrix ($\mathbf{A}$) and an $M \times 1$ vector of potential variables ($\boldsymbol{\psi}$) as follows.

$$\mathbf{A}(\boldsymbol{\alpha}_\xi)_{N\times M} \cdot \boldsymbol{\psi}_{M\times 1} \geq \mathbf{1}_{N\times 1}$$

The auxiliary value variables are represented as an array of vectors $\{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_k\}$ where each $2^m \times 1$ dimension auxiliary value vector is defined as follows.

$$(\boldsymbol{\alpha}_\xi)_{2^m\times 1} = \begin{bmatrix} \alpha_{\xi,1} \\ \alpha_{\xi,2} \\ \vdots \\ \alpha_{\xi,2^m} \end{bmatrix}$$

Using the example 1-bit XOR from above with $(m,n,l) = (2,2,5)$ and $(K,M,N) = (4,7,8)$, the values of the constraint matrix and potential vector are as follows.

$$\mathbf{A}(\boldsymbol{\alpha}_\xi)_{8\times 7} = \begin{bmatrix} 1 & 0 & -1 & 0 & -1 & 0 & \alpha_{1,1} \\ 1 & -\alpha_{1,1} & -1 & \alpha_{1,1} & -1 & \alpha_{1,1} & 0 \\ -1 & 0 & 1 & 0 & -1 & 0 & -\alpha_{1,2} \\ -1 & -\alpha_{1,2} & 1 & \alpha_{1,2} & -1 & -\alpha_{1,2} & 0 \\ -1 & 0 & -1 & 0 & 1 & 0 & -\alpha_{1,3} \\ -1 & -\alpha_{1,3} & -1 & -\alpha_{1,3} & 1 & \alpha_{1,3} & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & \alpha_{1,4} \\ 1 & -\alpha_{1,4} & 1 & -\alpha_{1,4} & 1 & -\alpha_{1,4} & 0 \end{bmatrix} \qquad \boldsymbol{\psi}_{7\times 1} = \begin{bmatrix} h_3 \\ h_4 \\ J_{1,3} \\ J_{1,4} \\ J_{2,3} \\ J_{2,4} \\ J_{3,4} \end{bmatrix}$$

Figure 4 below demonstrates how the constraint satisfaction problem scales for different operations and input sizes. Notice that there will be 2^K possible different configurations of the K auxiliary value variables since each can have a value of $+1$ or -1.

Operation	Spin Combination	In Spins (m)	Out Spins (n)	Aux Spins (k)	Potential Variables (M)	Auxiliary Variables (K)	Inequality Constraints (N)
XOR	1x1x1	2	1	1	7	4	8
	2x2x2	4	2	2	26	32	192
	3x3x3	6	3	3	57	192	3,584
	4x4x4	8	4	4	100	1,024	61,440
ADD	1x1x1	2	2	1	12	4	24
	2x2x2	4	3	2	35	32	448
	3x3x3	6	4	3	70	192	7,680
	4x4x4	8	5	4	117	1,024	126,976
MUL	2x2x1	4	4	1	35	16	480
	2x3x1	5	5	1	51	32	1,984
	2x4x2	6	6	2	84	128	16,128
	3x3x3	6	6	3	99	192	32,256
	2x5x2	7	7	2	108	256	65,024
	3x4x5	7	7	5	162	640	520,192
	4x4x8	8	8	8	264	2,048	16,711,680

FIGURE 4. A table of select operations and spin combinations demonstrating the scaling of the associated problem parameters.

STORING CONSTRAINTS AS COEFFICIENT MATRICES

The auxiliary constraint matrix ($\mathbf{A}$) from the previous section contains elements that are constant, ones that are linear in $\alpha_{i,j}$, and ones that are products of two values of $\alpha_{i,j}$. For convenience, coefficients of these terms are stored in *coefficient matrices* of the constant ($\mathbf{C}$), linear ($\mathbf{L}$), and quadratic ($\mathbf{Q}$) terms of $\mathbf{A}$. The coefficient matrices are generated and written to separate files by the program `genCoefMat.c` and read in by `genIsingData.py` to construct the auxiliary constraint matrix for a given set of auxiliary variables as follows.

$$\mathbf{A}(\boldsymbol{\alpha}_\xi)_{N\times M} = \mathbf{C}_{N\times M} + \sum_{i=1}^{k} \mathrm{diag}(\boldsymbol{\alpha}_i)_{N\times N} \cdot (\mathbf{L}_i)_{N\times M} + \sum_{i=1}^{k}\sum_{j=i+1}^{k} \mathrm{diag}(\boldsymbol{\alpha}_i \circ \boldsymbol{\alpha}_j)_{N\times N} \cdot (\mathbf{Q}_{i,j})_{N\times M}$$

The constraint matrices are all stored in *matrix market* (MM) format which records the row index, column index, and value of each non-zero entry. For the constant matrices, the number of non-zeros is low so the advantage of compressing the matrices is minimal. However, for the linear and quadratic coefficient matrices, the number of non-zeros decreases proportional to the total number of spins and the square of the total number of spins, so compression is much more efficient at conserving storage space. The number of non-zero elements for each of the coefficient matrices are given by the following.

$$\mathbf{C}_{nnz} = \frac{2^{n-1}}{2^n - 1}\left((m+n)(m+n+1) - m(m+1)\right)\frac{N}{2}$$

$$\mathbf{L}_{nnz} = (m+n+1)\frac{N}{2}$$

$$\mathbf{Q}_{nnz} = \frac{N}{2}$$

The *sparsity* of a matrix is the number of non-zero elements divided by the total number of elements. Therefore, the sparsity of the auxiliary constraint matrix ($\mathbf{A}$) is computed as follows. Notice that as the number of spins increases, sparsity asymptotically approaches one-half, meaning that sparse matrix compression overall saves about 2x in storage.

$$\mathbf{A}_{sp} = \frac{1}{MN}\left(\mathbf{C}_{nnz} + k\,\mathbf{L}_{nnz} + \frac{1}{2}k(k-1)\,\mathbf{Q}_{nnz}\right) \approx \frac{1}{2} \quad \text{for} \quad n \gg 1$$

A FINAL EXAMPLE: 2-BIT LOGICAL XOR WITH TWO AUXILIARY SPINS

For the final example consider a 2-bit logical XOR operation, which happens to be the logic unit with the fewest total spins that requires two auxiliary spins. An illustration can be found in Figure 5. The truth table for 2-bit logical XOR is as follows.

$$
\begin{aligned}
00 \oplus 00 &= 00 \qquad & 00 \oplus 01 &= 01 \\
01 \oplus 00 &= 10 \qquad & 01 \oplus 01 &= 11 \\
00 \oplus 10 &= 01 \qquad & 00 \oplus 11 &= 00 \\
01 \oplus 10 &= 11 \qquad & 01 \oplus 11 &= 10 \\
10 \oplus 00 &= 10 \qquad & 10 \oplus 01 &= 11 \\
11 \oplus 00 &= 00 \qquad & 11 \oplus 01 &= 01 \\
10 \oplus 10 &= 11 \qquad & 10 \oplus 11 &= 10 \\
11 \oplus 10 &= 01 \qquad & 11 \oplus 11 &= 00
\end{aligned}
$$

2-bit XOR Truth Table

s_1	s_2	s_3	s_4	s_5	s_6	a_1	a_2
-1	-1	-1	-1	-1	-1	± 1	± 1
-1	-1	-1	$+1$	-1	$+1$	± 1	± 1
-1	-1	$+1$	-1	$+1$	-1	± 1	± 1
-1	-1	$+1$	$+1$	$+1$	$+1$	± 1	± 1
-1	$+1$	-1	-1	-1	$+1$	± 1	± 1
-1	$+1$	-1	$+1$	-1	-1	± 1	± 1
-1	$+1$	$+1$	-1	$+1$	$+1$	± 1	± 1
-1	$+1$	$+1$	$+1$	$+1$	-1	± 1	± 1
$+1$	-1	-1	-1	$+1$	-1	± 1	± 1
$+1$	-1	-1	$+1$	$+1$	$+1$	± 1	± 1
$+1$	-1	$+1$	-1	-1	-1	± 1	± 1
$+1$	-1	$+1$	$+1$	-1	$+1$	± 1	± 1
$+1$	$+1$	-1	-1	$+1$	$+1$	± 1	± 1
$+1$	$+1$	-1	$+1$	$+1$	-1	± 1	± 1
$+1$	$+1$	$+1$	-1	-1	$+1$	± 1	± 1
$+1$	$+1$	$+1$	$+1$	-1	-1	± 1	± 1

Translate the truth table to Ising spins as before by letting spin down correspond to 0 or False, and spin up to 1 or True. The truth table in terms of Ising spins where s_1 through s_4 are input spins, s_5 and s_6 are output spins, and a_1 and a_2 are auxiliary spins is shown on the right.

The constraints for this new system with the auxiliary spin require that the energy of at least one correct configuration for each combination of input spins must be less than the energy of all the incorrect configurations corresponding to those input spins. Since there are $2^2 = 4$ auxiliary states, $2^4 = 16$ input states, and $2^2 = 4$ output status, there will be $4 \times 16 \times 3 = 192$ constraints. They will look like the following, presenting the first and last twelve inequalities only.

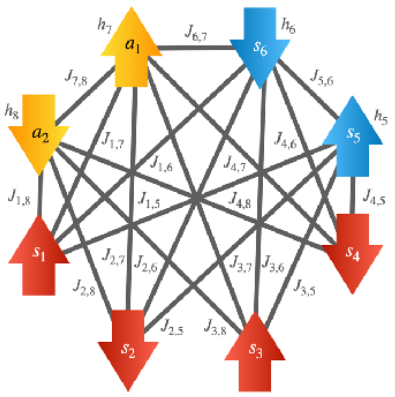


FIGURE 5. A 2-bit logical XOR consisting of four input spins (red), two output spin (blue), and two auxiliary spin (yellow).

$$
\begin{aligned}
&H(-1,-1,-1,-1,-1,+1,+\alpha_{1,1},+\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,-1,+1,+\alpha_{1,1},-\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,-1,+1,-\alpha_{1,1},+\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,-1,+1,-\alpha_{1,1},-\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,+1,-1,+\alpha_{1,1},+\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,+1,-1,+\alpha_{1,1},-\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,+1,-1,-\alpha_{1,1},+\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,+1,-1,-\alpha_{1,1},-\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,+1,+1,+\alpha_{1,1},+\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,+1,+1,+\alpha_{1,1},-\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,+1,+1,-\alpha_{1,1},+\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&H(-1,-1,-1,-1,+1,+1,-\alpha_{1,1},-\alpha_{2,1}) - H(-1,-1,-1,-1,-1,-1,+\alpha_{1,1},+\alpha_{2,1}) \geq 1\\
&\qquad\vdots\\
&H(-1,-1,-1,+1,-1,-1,+\alpha_{1,16},+\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,-1,-1,+\alpha_{1,16},-\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,-1,-1,-\alpha_{1,16},+\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,-1,-1,-\alpha_{1,16},-\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,+1,-1,+\alpha_{1,16},+\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,+1,-1,+\alpha_{1,16},-\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,+1,-1,-\alpha_{1,16},+\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,+1,-1,-\alpha_{1,16},-\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,+1,+1,+\alpha_{1,16},+\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,+1,+1,+\alpha_{1,16},-\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,+1,+1,-\alpha_{1,16},+\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1\\
&H(-1,-1,-1,+1,+1,+1,-\alpha_{1,16},-\alpha_{2,16}) - H(-1,-1,-1,+1,-1,-1,+1,+\alpha_{1,16},+\alpha_{2,16}) \geq 1
\end{aligned}
$$

The constraint satisfaction problem is a mixed integer cubic system of inequalities is abbreviated as follows. Notice the last column contains terms that are quadratic in $\alpha_{i,j}$ which causes the system to be cubic when dotted with the potential variable vector.

$$
\begin{bmatrix}
0 & 1 & -\alpha_{1,1} & -\alpha_{2,1} & 0 & -1 & \alpha_{1,1} & \alpha_{2,1} & 0 & -1 & \alpha_{1,1} & \alpha_{2,1} & 0 & -1 & \alpha_{1,1} & \alpha_{2,1} & 0 & -1 & \alpha_{1,1} & \alpha_{2,1} & -1 & \alpha_{1,1} & \alpha_{2,1} & 0 & 0 & 0\\
0 & 1 & -\alpha_{1,1} & 0 & 0 & -1 & \alpha_{1,1} & 0 & 0 & -1 & \alpha_{1,1} & 0 & 0 & -1 & \alpha_{1,1} & 0 & 0 & -1 & \alpha_{1,1} & 0 & -1 & \alpha_{1,1} & 0 & 0 & \alpha_{2,1} & -\alpha_{1,1}\alpha_{2,1}\\
0 & 1 & 0 & -\alpha_{2,1} & 0 & -1 & 0 & \alpha_{2,1} & 0 & -1 & 0 & \alpha_{2,1} & 0 & -1 & 0 & \alpha_{2,1} & 0 & -1 & 0 & \alpha_{2,1} & -1 & 0 & \alpha_{2,1} & \alpha_{1,1} & 0 & -\alpha_{1,1}\alpha_{2,1}\\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & \alpha_{1,1} & \alpha_{2,1} & 0\\
1 & 0 & -\alpha_{1,1} & -\alpha_{2,1} & -1 & 0 & \alpha_{1,1} & \alpha_{2,1} & -1 & 0 & \alpha_{1,1} & \alpha_{2,1} & -1 & 0 & \alpha_{1,1} & \alpha_{2,1} & -1 & 0 & \alpha_{1,1} & \alpha_{2,1} & -1 & 0 & 0 & \alpha_{1,1} & \alpha_{2,1} & 0\\
1 & 0 & -\alpha_{1,1} & 0 & -1 & 0 & \alpha_{1,1} & 0 & -1 & 0 & \alpha_{1,1} & 0 & -1 & 0 & \alpha_{1,1} & 0 & -1 & 0 & \alpha_{1,1} & 0 & -1 & 0 & \alpha_{2,1} & \alpha_{1,1} & 0 & -\alpha_{1,1}\alpha_{2,1}\\
1 & 0 & 0 & -\alpha_{2,1} & -1 & 0 & 0 & \alpha_{2,1} & -1 & 0 & 0 & \alpha_{2,1} & -1 & 0 & 0 & \alpha_{2,1} & -1 & 0 & 0 & \alpha_{2,1} & -1 & \alpha_{1,1} & 0 & 0 & \alpha_{2,1} & -\alpha_{1,1}\alpha_{2,1}\\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & \alpha_{1,1} & \alpha_{2,1} & 0 & 0 & 0\\
1 & 1 & -\alpha_{1,1} & -\alpha_{2,1} & -1 & -1 & \alpha_{1,1} & \alpha_{2,1} & -1 & -1 & \alpha_{1,1} & \alpha_{2,1} & -1 & -1 & \alpha_{1,1} & \alpha_{2,1} & -1 & -1 & \alpha_{1,1} & \alpha_{2,1} & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 1 & -\alpha_{1,1} & 0 & -1 & -1 & \alpha_{1,1} & 0 & -1 & -1 & \alpha_{1,1} & 0 & -1 & -1 & \alpha_{1,1} & 0 & -1 & -1 & \alpha_{1,1} & 0 & 0 & 0 & \alpha_{2,1} & 0 & \alpha_{2,1} & -\alpha_{1,1}\alpha_{2,1}\\
1 & 1 & 0 & -\alpha_{2,1} & -1 & -1 & 0 & \alpha_{2,1} & -1 & -1 & 0 & \alpha_{2,1} & -1 & -1 & 0 & \alpha_{2,1} & -1 & -1 & 0 & \alpha_{2,1} & 0 & \alpha_{1,1} & 0 & \alpha_{1,1} & 0 & -\alpha_{1,1}\alpha_{2,1}\\
1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & \alpha_{1,1} & \alpha_{2,1} & \alpha_{1,1} & \alpha_{2,1} & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
0 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & 0 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & 0 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & 0 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & 0 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & -1 & \alpha_{1,16} & \alpha_{2,16} & 0 & 0 & 0\\
0 & 1 & -\alpha_{1,16} & 0 & 0 & 1 & -\alpha_{1,16} & 0 & 0 & 1 & -\alpha_{1,16} & 0 & 0 & 1 & -\alpha_{1,16} & 0 & 0 & 1 & -\alpha_{1,16} & 0 & -1 & \alpha_{1,16} & 0 & 0 & \alpha_{2,16} & -\alpha_{1,16}\alpha_{2,16}\\
0 & 1 & 0 & -\alpha_{2,16} & 0 & 1 & 0 & -\alpha_{2,16} & 0 & 1 & 0 & -\alpha_{2,16} & 0 & 1 & 0 & -\alpha_{2,16} & 0 & 1 & 0 & -\alpha_{2,16} & -1 & 0 & \alpha_{2,16} & \alpha_{1,16} & 0 & -\alpha_{1,16}\alpha_{2,16}\\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & \alpha_{1,16} & \alpha_{2,16} & 0\\
1 & 0 & -\alpha_{1,16} & -\alpha_{2,16} & 1 & 0 & -\alpha_{1,16} & -\alpha_{2,16} & 1 & 0 & -\alpha_{1,16} & -\alpha_{2,16} & 1 & 0 & -\alpha_{1,16} & -\alpha_{2,16} & 1 & 0 & -\alpha_{1,16} & -\alpha_{2,16} & -1 & 0 & 0 & \alpha_{1,16} & \alpha_{2,16} & 0\\
1 & 0 & -\alpha_{1,16} & 0 & 1 & 0 & -\alpha_{1,16} & 0 & 1 & 0 & -\alpha_{1,16} & 0 & 1 & 0 & -\alpha_{1,16} & 0 & 1 & 0 & -\alpha_{1,16} & 0 & -1 & 0 & \alpha_{2,16} & \alpha_{1,16} & 0 & -\alpha_{1,16}\alpha_{2,16}\\
1 & 0 & 0 & -\alpha_{2,16} & 1 & 0 & 0 & -\alpha_{2,16} & 1 & 0 & 0 & -\alpha_{2,16} & 1 & 0 & 0 & -\alpha_{2,16} & 1 & 0 & 0 & -\alpha_{2,16} & -1 & \alpha_{1,16} & 0 & 0 & \alpha_{2,16} & -\alpha_{1,16}\alpha_{2,16}\\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & \alpha_{1,16} & \alpha_{2,16} & 0 & 0 & 0\\
1 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & 1 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & 1 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & 1 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & 1 & 1 & -\alpha_{1,16} & -\alpha_{2,16} & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 1 & -\alpha_{1,16} & 0 & 1 & 1 & -\alpha_{1,16} & 0 & 1 & 1 & -\alpha_{1,16} & 0 & 1 & 1 & -\alpha_{1,16} & 0 & 1 & 1 & -\alpha_{1,16} & 0 & 0 & 0 & \alpha_{2,16} & 0 & \alpha_{2,16} & -\alpha_{1,16}\alpha_{2,16}\\
1 & 1 & 0 & -\alpha_{2,16} & 1 & 1 & 0 & -\alpha_{2,16} & 1 & 1 & 0 & -\alpha_{2,16} & 1 & 1 & 0 & -\alpha_{2,16} & 1 & 1 & 0 & -\alpha_{2,16} & 0 & \alpha_{1,16} & 0 & \alpha_{1,16} & 0 & -\alpha_{1,16}\alpha_{2,16}\\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \alpha_{1,16} & \alpha_{2,16} & \alpha_{1,16} & \alpha_{2,16} & 0
\end{bmatrix}
\cdot
\begin{bmatrix}
h_5\\ h_6\\ h_7\\ h_8\\ J_{1,5}\\ J_{1,6}\\ J_{1,7}\\ J_{1,8}\\ J_{2,5}\\ J_{2,6}\\ J_{2,7}\\ J_{2,8}\\ J_{3,5}\\ J_{3,6}\\ J_{3,7}\\ J_{3,8}\\ J_{4,5}\\ J_{4,6}\\ J_{4,7}\\ J_{4,8}\\ J_{5,6}\\ J_{5,7}\\ J_{5,8}\\ J_{6,7}\\ J_{6,8}\\ J_{7,8}
\end{bmatrix}
\geq
\begin{bmatrix}
1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1
\end{bmatrix}
$$

SOLVING FOR FEASIBLE AUXILIARY VALUES: THE "ARTIFICIAL-INCREMENTAL" ALGORITHM

The general constraint satisfaction problem was generalized above as a simple system of inequality constraints.

$$\mathbf{A}(\boldsymbol{\alpha}_{\xi})_{N\times M}\cdot\boldsymbol{\psi}_{M\times 1}\geq\mathbf{1}_{N\times 1}$$

In order to satisfy this set of constraints it was necessary to choose an auxiliary value vector $(\boldsymbol{\alpha}_{\xi})_{2^m\times 1}$ and then solve for a potential vector $\boldsymbol{\psi}_{M\times 1}$. When the constraints are satisfiable for a given choice of the auxiliary value vector it is said to be feasible. This section discusses the current best method of identifying feasible auxiliary vector for a given constraint problem using *artificial variables*.

An artificial variable is a type of variable introduced into a constraint satisfaction problem to test for a basic feasible solution. It is called "artificial" because it does not correspond to any physical or real characteristics of the problem. Let $\boldsymbol{\rho}_{N\times 1}$ be a vector of N artificial variables added to the constraint satisfaction problem to form a new set of constraints.

$$\mathbf{A}(\boldsymbol{\alpha}_{\xi})_{N\times M}\cdot\boldsymbol{\psi}_{M\times 1}+\boldsymbol{\rho}_{N\times 1}\geq\mathbf{1}_{N\times 1}$$

Notice that for the new constraint satisfaction problem it is *always* possible to choose values of the artificial variables that satisfy the inequalities. Additionally, in the case that the $\boldsymbol{\rho}_{N\times 1}$ are all zero the solutions $(\boldsymbol{\alpha}_{\xi})_{2^m\times 1}$ and $\boldsymbol{\psi}_{M\times 1}$ to the new problem will also be solutions to the original problem. The sum of the artificial variables is therefore an objective that measures how far a given auxiliary value vector is from creating a satisfiable system of inequalities. When the objective is zero the original system will be satisfiable.

The *artificial-incremental algorithm* minimizes the sum of the artificial variables over a sequence of incremental updates to the auxiliary value vector.

$$\min_{\boldsymbol{\rho}\in\mathbb{R}^N}\sum_{i=1}^{N}\rho_i\quad\text{with}\quad\mathbf{A}(\boldsymbol{\alpha}_{\xi})_{N\times M}\cdot\boldsymbol{\psi}_{M\times 1}+\boldsymbol{\rho}_{N\times 1}\geq\mathbf{1}_{N\times 1}\quad\text{and}\quad\boldsymbol{\rho}_{N\times 1}>\mathbf{0}_{N\times 1}$$

The artificial-incremental algorithm proceeds as follows:

1. Start with a randomly generated initial auxiliary value vector $\boldsymbol{\alpha}_{\xi}^{(0)}$ assuming a single auxiliary spin, i.e., $k = 1$.
2. Construct the $K - 1$ candidate auxiliary value vectors that each differ from $\boldsymbol{\alpha}_{\xi}^{(0)}$ in exactly one of its K dimension.
3. Solve the minimization problem for each candidate auxiliary value vectors using a simple linear programming solver.
4. Chose $\boldsymbol{\alpha}_{\xi}^{(1)}$ to be the auxiliary value vector that minimizes the objective. If there is more than one minimum then choose at random from the available minima.
5. Repeat until the minimum objective stops decreasing.
    a. If the minimum objective is greater than zero add a new auxiliary spin, i.e., $k = k + 1$, and restart the algorithm at step #2 above.
        i. *Random* — Initialize the new spin to a random value.
        ii. P*revious* — Initialize the new spin to the final values of the previous spin.
    b. If the minimum objective is greater than zero and the number of auxiliary spins exceeds a user defined upper limit then restart the algorithm at step #1 above.
    c. If the minimum objective is zero the auxiliary value vector is feasible.

Remember that a random search of the auxiliary value vector space scales exponentially and could require generating as many as 2^K candidate solutions. On the other hand, the artificial-incremental algorithm scales polynomially in K making it significantly more efficient than random search.

In practice, the artificial-incremental algorithm converges relatively quickly for up to 3-bit multiplication, but the solve time is highly dependent on the initial choice of $\boldsymbol{\alpha}_{\xi}^{(0)}$. With a good initial value vector the solution will converge rapidly, but with a poor initial value vector the solution will converge slowly and may require multiple restarts. Over 100 runs of the algorithm to compute MUL_3x3x3, the minimum solve time was 4.6 minutes and the maximum solve time was 17.5 hours. The median solve time was 47 minutes.