Natural Computing
https://doi.org/10.1007/s11047-021-09845-3

Solving combinatorial optimisation problems using oscillator based Ising machines

Tianshi Wang[1] · Leon Wu[1] · Parth Nobel[1] · Jaijeet Roychowdhury[1]

Accepted: 16 January 2021

Abstract
We present OIM (Oscillator Ising Machines), a new way to make Ising machines using networks of coupled self-sustaining nonlinear oscillators. OIM is theoretically rooted in a novel result that establishes that the phase dynamics of coupled oscillator systems, under the influence of subharmonic injection locking, are governed by a Lyapunov function that is closely related to the Ising Hamiltonian of the coupling graph. As a result, the dynamics of such oscillator networks evolve naturally to local minima of the Lyapunov function. Two simple additional steps (i.e., turning subharmonic locking on and off smoothly, and adding noise) enable the network to find excellent solutions of Ising problems. We demonstrate our method on Ising versions of the MAX-CUT and graph colouring problems, showing that it improves on previously published results on several problems in the G benchmark set. Using synthetic problems with known global minima, we also present initial scaling results. Our scheme, which is amenable to realisation using many kinds of oscillators from different physical domains, is particularly well suited for CMOS IC implementation, offering significant practical advantages over previous techniques for making Ising machines. We report working hardware prototypes using CMOS electronic oscillators.

Keywords Ising machines · Oscillators · CMOS · Hamiltonian · Lyapunov · MAX-CUT · Frustrated loops · Graph colouring

1 Introduction

The Ising model (Ising 1925; Brush 1967) takes any weighted graph and uses it to define a scalar function called the Ising Hamiltonian. Each vertex in the graph is associated with a *spin*, i.e., a binary variable taking values ± 1. The Ising problem is to find an assignment of spins that minimises the Ising Hamiltonian (which depends on the spins and on the graph's weights). Solving the Ising problem in general has been shown to be very difficult (Barahona 1982), but devices that can solve it quickly using specialised hardware have been proposed in recent years (Marandi et al. 2014; McMahon et al. 2016; Inagaki et al. 2016; Johnson et al. 2011; Bian et al. 2014; Yamaoka et al. 2016). Such Ising machines have attracted much interest because many classically difficult combinatorial optimisation problems (including all 21 of Karp's well-known list of NP-complete problems Karp 1972) have known mappings to Ising problems (Lucas 2014).[1] Hence, as Moore's Law nears its limits, Ising machines offer promise as a novel alternative paradigm for solving difficult computational problems efficiently.

We present a new and attractive means for realising Ising machines, i.e., using networks of coupled, self-sustaining nonlinear oscillators. We first establish a key theoretical result that relates the (continuous) phase dynamics

✉ Jaijeet Roychowdhury
jr@berkeley.edu

Tianshi Wang
tianshi@berkeley.edu

Leon Wu
eon@berkeley.edu

Parth Nobel
parthnobel@berkeley.edu

[1] Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA

[1] Indeed, since the Ising Hamiltonian ground state problem is NP-complete (Barahona 1982), *every* NP-complete problem can in principle be mapped to Ising.

Published online: 05 May 2021

of an oscillator network with the (discrete/combinatorial) Ising Hamiltonian of the graph representing the oscillator couplings (Wang and Roychowdhury 2017). We then build on this result to develop practical oscillator-based Ising machines and demonstrate their effectiveness by solving MAX-CUT and graph colouring combinatorial optimisation problems (Festa et al. 2002; Jensen and Toft 1995). We also present working hardware prototypes of our oscillator-based Ising machines.

We first show that the phase dynamics of any network of coupled, self-sustaining, amplitude-stable oscillators can be abstracted using the Generalised Adler model (Gen-Adler) (Neogy and Roychowdhury 2012; Bhansali and Roychowdhury 2009), which leads to a generalisation of the well-known Kuramoto model (Kuramoto 1975, 2003; Acebrón et al. 2005). The model's phase dynamics are governed by an associated Lyapunov function, i.e., a scalar function of the oscillators' phases that is always non-increasing and settles to stable local minima as phase dynamics evolve. If each oscillator's phase settles to either 0 or π (radians) and these values are associated with spins of ± 1, we show that this Lyapunov function is essentially identical to the Ising Hamiltonian of the oscillator network's connectivity graph. In general, however, oscillator phases do not settle to the discrete values $0/\pi$, but span a continuum of values. In order to binarise oscillator phases (i.e., get them to settle to values near $0/\pi$), we inject each oscillator with a second harmonic signal (dubbed SYNC) that induces subharmonic injection locking (SHIL), which makes the phase of each oscillator settle to a value near either 0 or π (Wang and Roychowdhury 2014; Neogy and Roychowdhury 2012; Wang 2017a, b). We devise a new Lyapunov function that governs the network's dynamics with SHIL, then show its equivalence to the Ising Hamiltonian at phase values of $0/\pi$.

Thus we show that when SHIL binarisation is applied, coupled oscillator network dynamics settle naturally to *local* minima of a continuised version of the associated Ising Hamiltonian. To evolve the system out of local minima towards a global minimum, we show that a simple scheme, in which the binarising second-harmonic SYNC signal's amplitude is ramped up and down together with judicious amounts of noise added, works well.[2]

Our approach, dubbed OIM (Oscillator Ising Machine) is analogous to several existing schemes to map a system's Lyapunov function to a minimisation objective, including the Hopfield–Tank neural network proposed for the travelling salesman problem (Hopfield and Tank 1985), and the more recent work on designing differential equations for satisfiability (SAT) problems (Ercsey-Ravasz and Toroczkai 2011; Yin et al. 2018). Based on the equivalence we establish (in Sect. 3.3, below) between the Ising Hamiltonian and a Lyapunov function for coupled oscillators under SHIL, OIM can be applied to problems that have an Ising formulation (Lucas 2014), a wide class that includes the problems addressed by these other schemes.

We present simulation results on a standard MAX-CUT benchmark set of 54 large problems, demonstrating that OIM finds the best-known results in many cases, indeed better results than seem to have been previously published for several of the problems. We also present initial results on so-called "frustrated loop" problems, i.e., synthetic problems that have known global minima. Such problems enable us to assess the probability of actually finding global minima. Our initial results, which are restricted to the particular type of frustrated loop problems used in Sheldon et al. (2019), indicate that the time taken to find global minima scales polynomially with problem sizes in the range 216 to 64000; but at the same time, we find that simulated annealing's performance on these problems is polynomial as well. As another application, we demonstrate OIM on the graph colouring problem. We summarize prototypes (with up to 240 spins) built on breadboard and PCB that function well, testifying to the ease with which practical hardware implementations can be built.

Our scheme is different from previous Ising machine approaches, which are of 3 types (see Sect. 2): (1) a fibre-optic laser-based scheme known as the Coherent Ising Machine (Marandi et al. 2014; McMahon et al. 2016; Inagaki et al. 2016), (2) the D-WAVE quantum Ising machine (Johnson et al. 2011; Bian et al. 2014) and (3) CMOS hardware-accelerated simulated annealing chips for solving Ising problems (Yamaoka et al. 2016; Aramon 2019; Gyoten et al. 2018a, b). Unlike CIM and D-WAVE, which are large, expensive and ill-suited to low-cost mass production, our approach is a purely classical scheme that does not rely on quantum phenomena or novel nano-devices. Indeed, it can be implemented using conventional CMOS electronics, which has many advantages: scalability/miniaturisability (i.e., very large numbers of spins in a physically small system), well-established design processes and tools that essentially guarantee first-time working hardware, very low power operation, seamless integration with control and I/O logic, easy programmability via standard interfaces like USB, and low-cost mass production. CMOS implementations of our scheme also allow flexibility in introducing controlled noise and programming SYNC ramping schedules. Furthermore, implementing oscillator coupling by physical connectivity makes our scheme inherently parallel, unlike CIM, where coupling is implemented via FPGA-based digital computation and is

[2] This is somewhat analogous to annealing schedules that are used in simulated annealing (SA, Myklebust 2015), though we stress that the underlying minimization mechanism of OIM is completely different from that of SA.

inherently serial. The advantages of CMOS also apply, of course, to hardware simulated annealing engines (Yamaoka et al. 2016; Aramon 2019; Gyoten et al. 2018a, b), but our scheme has additional attractive features. One key advantage relates to variability, a significant problem in nanoscale CMOS. For oscillator networks, device- and circuit-level variability impacts the system by causing a spread in the natural frequencies of the oscillators. Unlike other schemes, where performance deteriorates due to variability (Yamaoka et al. 2016), we can essentially eliminate variability by means of simple VCO-based calibration to bring all the oscillators to the same frequency.[3] Another potential advantage stems from the continuous/analog nature of our scheme (as opposed to purely digital simulated annealing schemes). Computational experiments indicate that the time our scheme takes to find good solutions of the Ising problem grows only very slowly with respect to the number of spins. This is a significant potential advantage over simulated annealing schemes (Aramon 2019) as hardware sizes scale up to large numbers of spins. Note that we can use virtually any type of nonlinear oscillator (not just CMOS) to implement our scheme, including optical, MEMS, biochemical, spin-based, etc., oscillators; however, CMOS seems the easiest and most advantageous implementation route given the current state of technology.

In the remainder of this paper, we first provide a brief summary of the Ising problem and existing Ising machine schemes in Sect. 2. We then present our oscillator-based Ising machine scheme (dubbed OIM, for Oscillator Ising Machine) in Sect. 3, explaining the theory that enables it to work. Then in Sect. 4, we present both computational and hardware examples showing the effectiveness of our scheme for solving several combinatorial optimisation problems.

2 The Ising problem and existing Ising machine approaches

The Ising model is named after the German physicist Ernst Ising. It was first studied in the 1920s as a mathematical model for explaining domain formation in ferromagnets (Ising 1925). It comprises a group of discrete variables $\{s_i\}$, dubbed *spins*, each taking a binary value ± 1, such that an associated "energy function", known as the Ising Hamiltonian, is minimised. The Ising Hamiltonian is defined as

$$H \triangleq -\sum_{1 \le i < j \le n} J_{ij} s_i s_j - \sum_{i=1}^{n} h_i s_i, \quad \text{such that } s_i \in \{-1, +1\}, \tag{1}$$

where n is the number of spins; $\{J_{ij}\}$ and $\{h_i\}$ are real coefficients. The Ising model is often simplified by dropping the $\{h_i\}$ terms. Under this simplification, the Ising Hamiltonian becomes

$$H = -\sum_{i,j,i<j} J_{ij} s_i s_j. \tag{2}$$

What makes the Ising model particularly interesting is that many hard optimisation problems can be shown to be equivalent to it (Barahona 1982; Bian et al. 2010). In fact, all of Karp's 21 NP-complete problems can be mapped to it by assigning appropriate values to the coefficients (Lucas 2014). Physical systems that can directly minimise the Ising Hamiltonian, namely Ising machines, thus become very attractive for potentially outperforming conventional algorithms run on CPUs for these problems.

Several schemes have been proposed recently for realising Ising machines in hardware. One well-known example is from D-Wave Systems (Johnson et al. 2011; Bian et al. 2014). Their quantum Ising machines use superconducting loops as spins and connect them using Josephson junction devices (Harris et al. 2010). As the machines require a temperature below 80 mK ($-273.07°$C) to operate (Johnson et al. 2011), they all have a large footprint to accommodate the necessary cooling system. While some question their advantages over simulated annealing run on classical computers (Rønnow et al. 2014), proponents believe that through a mechanism known as quantum annealing, they can offer large speedups on problems with certain energy landscapes (Denchev et al. 2016).

Other proposals use novel non-quantum devices as Ising spins instead, so that the machines can function at room temperature. Most notable among them is a scheme based on lasers and long optical fibres (Marandi et al. 2014; McMahon et al. 2016; Inagaki et al. 2016). The Ising spins are represented using time-multiplexed optical parametric oscillators (OPOs), which are laser pulses travelling on the same fibre. The coupling between these pulses is implemented digitally by measurement and feedback using FPGA chips. While these machines can potentially be more compact than D-Wave's machines, it is unclear how far they can be miniaturised and integrated. Recent studies have also proposed the use of several novel nanodevices as Ising spins, including MEMS (Micro-Electro-Mechanical Systems) resonators (Mahboob et al. 2016) and nanomagnets from Spintronics (Camsari et al. 2017). Physical realisation of these machines still awaits future development of these emerging device technologies.

[3] Moreover, as we show in Sect. 3.4, our scheme is inherently resistant to variability even without such calibration.

Another broad direction is to build Ising model emulators using digital circuits. A recent implementation (Yamaoka et al. 2016) uses CMOS SRAM cells as spins, and couples them using digital logic gates. The authors point out, however, that "the efficacy in achieving a global energy minimum is limited" (Yamaoka et al. 2016) due to variability. The speedup and accuracy reported by Yamaoka et al. (2016) are instead based on deterministic on-chip computation paired with an external random number generator—a digital hardware implementation of the simulated annealing algorithm. More recently, similar digital accelerators have also been tried on FPGAs (Yamamoto et al. 2017). These implementations are not directly comparable to the other Ising machine implementations discussed above, which attempt to use interesting intrinsic physics to minimise the Ising Hamiltonian while achieving large speedups.

3 Oscillator-based Ising machines

In this section, we show that a network of coupled self-sustaining oscillators can function as an Ising machine. To do so, we first study the response of a single oscillator under injection locking in Sect. 3.1. Specifically, we examine the way the oscillator's phase locks to that of the external input. While regular injection locking typically aligns the oscillator's phase with the input, as illustrated in Fig. 1a and b, its variant—subharmonic injection locking (SHIL)—can make the oscillator develop multiple stable phase-locked states (Fig. 1c and d). As we show in Sect. 3.1, these phenomena can be predicted accurately using the Gen-Adler model (Bhansali and Roychowdhury 2009).

The Gen-Adler equation of a single oscillator, when extended to the phase dynamics of coupled oscillator networks, becomes equivalent to a variant of the Kuramoto model. In Sect. 3.2, we show that the model's dynamics are governed by a global Lyapunov function, a scalar "energy-like" quantity that is naturally minimised by the coupled oscillator network. Then in Sect. 3.3, we introduce SHIL into the system to binarise the phases of oscillators. As illustrated in Fig. 1e, SHIL induces each oscillator to settle to one of two stable phase-locked states. Due to the coupling between them, a network of such binarised oscillators will prefer certain phase configurations over others. We confirm this intuition in Sect. 3.3 by deriving a new Lyapunov function that such a system (i.e., with SHIL) minimises. By examining this function's equivalence to the Ising Hamiltonian, we show that such a coupled oscillator network under SHIL indeed physically implements an Ising machine. Finally, in Sect. 3.4, we consider the effect of variability on the system's operation. We show that a spread in the natural frequencies of the oscillators contributes a linear term in the global Lyapunov function, which does not affect Ising machine performance by much if the variability is not extreme.

3.1 Injection locking in oscillators

When an oscillator with natural frequency ω_0 is perturbed by a small periodic input at a similar frequency ω_1, its phase response can be predicted well by the Generalised Adler model (Gen-Adler) (Bhansali and Roychowdhury 2009). Gen-Adler has the following form:

$$\frac{d}{dt}\phi(t) = \omega_0 - \omega_1 + \omega_0 \cdot c(\phi(t) - \phi_{in}), \tag{3}$$

where $\phi(t)$ and ϕ_{in} are the phases of the oscillator and the perturbation. $c(\cdot)$ is a 2π-periodic function derived based on

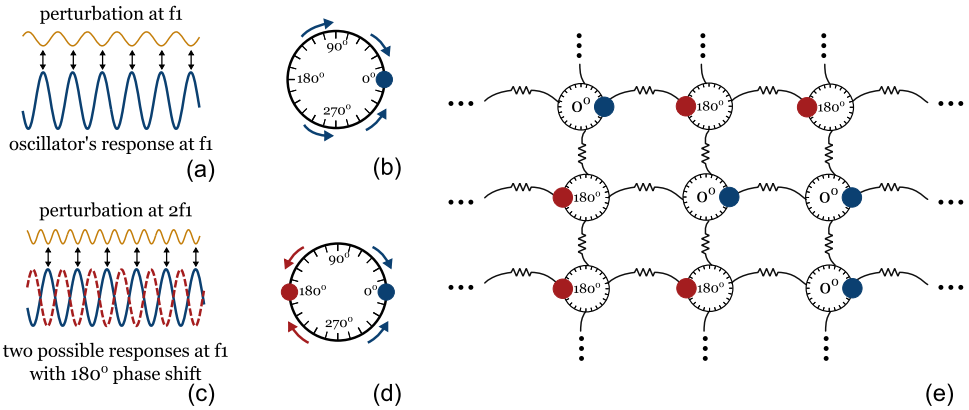


Fig. 1 Illustration of the basic mechanism of oscillator-based Ising machines: (**a**) oscillator shifts its natural frequency from f_0 to f_1 under external perturbation; (**b**) oscillator's phase becomes stably locked to the perturbation; (**c**) when the perturbation is at $2f_1$, the oscillator locks to its subharmonic at f_1; (**d**) bistable phase locks under subharmonic injection locking; (**e**) coupled subharmonically injection-locked oscillators settle with binary phases representing an optimal spin configuration for an Ising problem

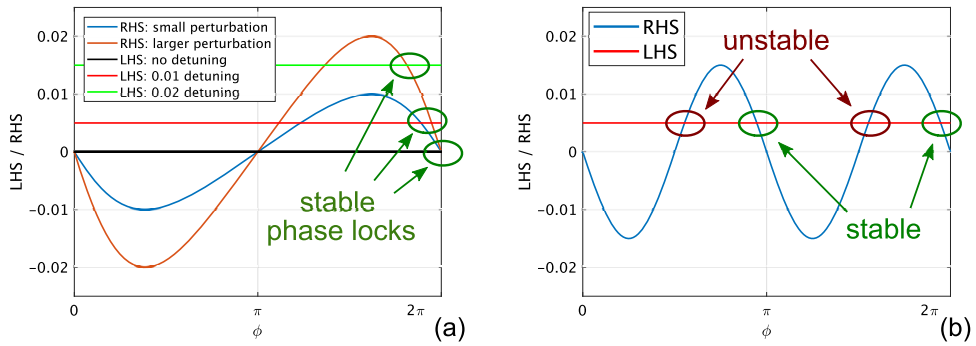


Fig. 2 Illustration of the LHS and RHS of the equilibrium Gen-Adler equation. (**a**) Under normal injection locking, the intersection of LHS and RHS predicts the only stable solution of ϕ under different scenarios. (**b**) Perturbation at $2\omega_1$ changes the shape of $c(.)$ in Gen-Adler; the intersections now predict the locations of two stable phase-locked states

an intrinsic property of the oscillator known as the Phase Response Curve (PRC) (Winfree 1967) or the Perturbation Projection Vector (PPV) (Demir et al. 2000). A derivation of Gen-Adler from the low-level differential equations of an oscillator is provided in a preprint of this paper (Wang and Roychowdhury 2019a).

The Gen-Adler equation governs the dynamics of the oscillator's phase under periodic inputs; its equilibrium states can be used to accurately predict the injection-locked states of the oscillator. The equilibrium Gen-Adler equation can be derived by rearranging (3):

$$\frac{\omega_1 - \omega_0}{\omega_0} = c(\phi^* - \phi_{in}), \tag{4}$$

where ϕ^* is the solution of phase in equilibrium.

The Left Hand Side (LHS) of (4) is a constant representing the frequency detuning of the oscillator from the input; the Right Hand Side (RHS) is a periodic function of ϕ^*; its magnitude depends on both the PPV of the oscillator and the strength of the input (Bhansali and Roychowdhury 2009). By plotting both terms and looking for intersections, one can easily predict whether injection locking will occur, and if it does, what the locked phase of the oscillator ϕ^* will be. Figure 2a plots a few examples of LHS/RHS, showing their shapes and magnitudes under different conditions.

As mentioned in Sect. 1, SHIL can occur when the external input is about twice as fast as the oscillator. When the input is at frequency $\omega_1 \simeq 2\omega_0$, it can be shown that the corresponding $c(\cdot)$ becomes a π-periodic function (Neogy and Roychowdhury 2012; Wang and Roychowdhury 2015a); a typical example is given in Fig. 2b, where $c(\cdot)$ takes the shape of $-\sin(2\phi)$. In this case, two of the four LHS-RHS intersections represent stable phase-locked states; it can be shown that they are separated by a phase difference of 180° (Neogy and Roychowdhury 2012). Gen-Adler is a powerful technique for predicting and understanding injection locking in oscillators and constitutes an important foundation for the analyses that follow.

3.2 Global Lyapunov function

For an oscillator in a coupled oscillator network, its external perturbations come from the other oscillators connected to it. Its Gen-Adler equation can be written as

$$\frac{d}{dt}\phi_i(t) = \omega_i - \omega^* + \omega_i \cdot \sum_{j=1,\, j\neq i}^{n} c_{ij}(\phi_i(t) - \phi_j(t)), \tag{5}$$

where $\{\phi_i\}$ represent the phases of n oscillators, ω_i is the frequency of the ith oscillator, and ω^* is the central frequency of the network. $c_{ij}(\cdot)$ is a 2π-periodic function that abstracts relevant properties of the coupling between oscillator i and oscillator j, as well as of the PPVs of the oscillators.

To simplify exposition, we now assume that the c_{ij} functions are sinusoidal, although in Wang and Roychowdhury (2019a), we show that this does not have to be the case for the analysis to hold true.[4] We further assume zero spread in the natural frequencies of oscillators, i.e., $\omega_i \equiv \omega^*$, and discuss the effect of frequency variability later in Sect. 3.4. With these simplifications, (5) can be written as

$$\frac{d}{dt}\phi_i(t) = -K \cdot \sum_{j=1,\, j\neq i}^{n} J_{ij} \cdot \sin(\phi_i(t) - \phi_j(t)). \tag{6}$$

Here, we are using the coefficients $\{J_{ij}\}$[5] from the Ising model (1) to set the connectivity of the network, i.e., the coupling strength between oscillators i and j is proportional to J_{ij}. $\omega_i \equiv \omega^*$ is incorporated in the parameter K, which modulates the overall coupling strength of the network.

There is a global Lyapunov function associated with (6) (Shinomoto and Kuramoto 1986):

[4] More generally, c_{ij}s can be any 2π-periodic odd functions, which are better suited to practical oscillators.

[5] In the Ising Hamiltonian (1), J_{ij} is only defined when $i<j$; here we assume that $J_{ij} = J_{ji}$ for all i, j.

$$E(\vec{\phi}(t)) = -K \cdot \sum_{i,j,\, i \neq j} J_{ij} \cdot \cos(\phi_i(t) - \phi_j(t)), \tag{7}$$

where $\vec{\phi}(t) = [\phi_1(t), \ldots, \phi_n(t)]^T$. Being a global Lyapunov function, it is an objective function the coupled oscillator system always tends to minimise as it evolves over time (Lyapunov 1992).

If we look at the values of this continuous function $E(\vec{\phi}(t))$ at some discrete points, we notice that it shares some similarities with the Ising Hamiltonian. At points where every ϕ_i is equal to either 0 or π,[6] if we map $\phi_i = 0$ to $s_i = +1$ and $\phi_i = \pi$ to $s_i = -1$, we have

$$\begin{aligned} E(\vec{\phi}(t)) &= -K \cdot \sum_{i,j,i \neq j} J_{ij} \cdot \cos(\phi_i(t) - \phi_j(t)) \\ &= -K \cdot \sum_{i,j,i \neq j} J_{ij} s_i s_j = -2K \cdot \sum_{i,j,i<j} J_{ij} s_i s_j. \end{aligned} \tag{8}$$

If we choose $K = 1/2$, the global Lyapunov function in (7) exactly matches the Ising Hamiltonian in (2) at these discrete points. But this does not mean that coupled oscillators are naturally minimising the Ising Hamiltonian, as there is no guarantee at all that the phases $\{\phi_i(t)\}$ are settling to these discrete points. In fact, networks with more than two oscillators almost always synchronise with analog phases, i.e., $\{\phi_i(t)\}$ commonly settle to continuous values spread out in the phase domain as opposed to converging towards 0 and π. As an example, Fig. 3a shows the phase responses of 20 oscillators connected in a random graph. As phases do not settle to the discrete points discussed above, the Lyapunov function they minimise becomes irrelevant to the Ising Hamiltonian, rendering the system ineffective for solving Ising problems. While one may think that the analog phases can still serve as solutions when rounded to the nearest discrete points, experiments in Sect. 4.3 show that the quality of these solutions is very poor compared with our scheme of Ising machines proposed in this paper.

3.3 Network of coupled oscillators under SHIL and its global Lyapunov function

In our scheme, a common SYNC signal at $2\omega^*$ is injected into every oscillator in the network. Through the mechanism of SHIL, the oscillator phases are binarised. The example shown in Fig. 3b confirms that this is indeed the case: under SHIL, the phases of 20 oscillators connected in the same random graph now settle very close to discrete points. Recall from Sect. 3.1 that a $\omega_1 \simeq 2\omega_0$ perturbation introduces a π-periodic coupling term (e.g., $\sin(2\phi)$) in the phase dynamics; (6) becomes (see Wang and Roychowdhury 2019a for the derivation)

$$\begin{aligned} \frac{d}{dt}\phi_i(t) = &-K \cdot \sum_{j=1,\, j \neq i}^{n} J_{ij} \cdot \sin(\phi_i(t) - \phi_j(t)) \\ &- K_s \cdot \sin(2\phi_i(t)), \end{aligned} \tag{9}$$

where K_s represents the strength of coupling from SYNC, i.e., the input signal at $\omega_1 \simeq 2\omega_0$.

Remarkably, there is a global Lyapunov function for (9) as well:

$$\begin{aligned} E(\vec{\phi}(t)) = &-K \cdot \sum_{i,j,\, i \neq j} J_{ij} \cdot \cos(\phi_i(t) - \phi_j(t)) \\ &- K_s \cdot \sum_{i=1}^{n} \cos(2\phi_i(t)). \end{aligned} \tag{10}$$

We now show that E in (10) is indeed a global Lyapunov function. To do so, we first differentiate E with respect to $\vec{\phi}$.

We observe that the first component of E is the sum of $(n^2 - n)$ number of cos() terms. Among them, for any given index k, the variable ϕ_k appears a total of $2 \cdot (n-1)$ times. It appears $(n-1)$ times as the subtrahend inside cos(): these $(n-1)$ terms are $J_{kl} \cdot \cos(\phi_k(t) - \phi_l(t))$, where $l = 1, \ldots, n$ and $l \neq k$. For the other $(n-1)$ times, it appears as the minuend inside cos(): in $J_{lk} \cdot \cos(\phi_l(t) - \phi_k(t))$, where $l = 1, \ldots, n$, $l \neq k$. So when we differentiate E with respect to ϕ_k, we have

$$\begin{aligned} \frac{\partial E(\vec{\phi}(t))}{\partial \phi_k(t)} &= -K \cdot \sum_{l=1,\, l \neq k}^{n} J_{kl} \frac{\partial}{\partial \phi_k(t)} [\cos(\phi_k(t) - \phi_l(t))] \\ &\quad - K \cdot \sum_{l=1,\, l \neq k}^{n} J_{lk} \frac{\partial}{\partial \phi_k(t)} [\cos(\phi_l(t) - \phi_k(t))] \\ &\quad - K_s \cdot \frac{\partial}{\partial \phi_k(t)} \cos(2\phi_k(t)) \\ &= K \cdot \sum_{l=1,\, l \neq k}^{n} J_{kl} \sin(\phi_k(t) - \phi_l(t)) \\ &\quad - K \cdot \sum_{l=1,\, l \neq k}^{n} J_{lk} \sin(\phi_l(t) - \phi_k(t)) \\ &\quad + K_s \cdot 2 \cdot \sin(2\phi_k(t)) \\ &= K \cdot \sum_{l=1,\, l \neq k}^{n} J_{kl} \cdot 2 \cdot \sin(\phi_k(t) - \phi_l(t)) \\ &\quad + K_s \cdot 2 \cdot \sin(2\phi_k(t)) \\ &\qquad (\text{using } \sin(x) = -\sin(-x) \text{ and } J_{lk} = J_{kl}) \\ &= -2 \cdot \frac{d\phi_k(t)}{dt}. \end{aligned} \tag{11}$$

Therefore,

[6] More generally, we can use $\{2k\pi \mid k \in \mathbf{Z}\}$ and $\{2k\pi + \pi \mid k \in \mathbf{Z}\}$ to represent the two states for each oscillator's phase.

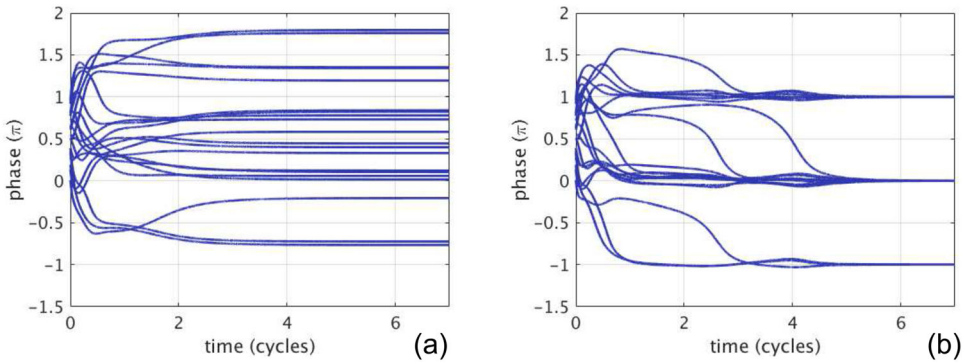


Fig. 3 Phases of 20 oscillators with random $\{J_{ij}\}$ generated by `rudy -rnd_graph 20 50 10001`: (**a**) without SYNC; (**b**) with $K_s = 1$

$$\frac{\partial E(\vec{\phi}(t))}{\partial t} = \sum_{k=1}^{n}\left[\frac{\partial E(\vec{\phi}(t))}{\partial \phi_k(t)} \cdot \frac{d\phi_k(t)}{dt}\right] = -2 \cdot \sum_{k=1}^{n}\left(\frac{d\phi_k(t)}{dt}\right)^2 \leq 0. \tag{12}$$

Thus, we have proved that (10) is indeed a global Lyapunov function which is naturally minimized over time by coupled oscillators under SHIL. A similar but more detailed proof for the general case, where we do not assume sinusoidal coupling functions, is given in Wang and Roychowdhury (2019a).

At the discrete points (phase values of $0/\pi$), because $\cos(2\phi_i) \equiv 1$, (10) reduces to

$$E(\vec{\phi}(t)) \approx -K \cdot \sum_{i,j,\, i \neq j} J_{ij} \cdot \cos(\phi_i(t) - \phi_j(t)) - n \cdot K_s, \tag{13}$$

where $n \cdot K_s$ is a constant. By choosing $K = 1/2$, we can then make (13) equivalent to the Ising Hamiltonian in (2) with a constant offset.

Note that the introduction of SYNC does not change the relative E levels between the discrete points, but modifies them by the *same* amount. However, with SYNC, all phases can be forced to eventually take values near either 0 or π—the system now tries to reach a binary state that minimises the Ising Hamiltonian, thus functioning as an Ising machine. We emphasise that this is *not* equivalent to running the system without SHIL and then rounding the analog phase solutions to discrete values as a post-processing step. Instead, the introduction of SHIL modifies the energy landscape of E, changes the dynamics of the coupled oscillator system, and as we show in Sect. 4, results in greatly improved minimisation of the Ising Hamiltonian.

It is worth noting, also, that the Lyapunov function in (10) will, in general, have many local minima and there is no guarantee the oscillator-based Ising machine will settle at or near any global optimal state. However, when judicious amounts of noise are introduced via a noise level parameter K_n, we have seen empirically that the system is more likely to settle to lower minima. Noise in the phases of oscillators is commonly modelled by adding white noise sources to the oscillator frequencies, changing the system Eq. (9) to

$$\frac{d}{dt}\phi_i(t) = -K \cdot \sum_{j=1,\, j\neq i}^{n} J_{ij} \cdot \sin(\phi_i(t) - \phi_j(t)) - K_s \cdot \sin(2\phi_i(t)) + K_n \cdot \xi_i(t), \tag{14}$$

where variable $\xi_i(t)$ represents Gaussian white noise with a zero mean and auto-correlation $\langle \xi_i(t), \xi_i(\tau) \rangle = \delta(t - \tau)$; the scalar K_n represents the magnitude of noise (Wang and Roychowdhury 2019a).

Indeed, the several parameters in the Ising machine—K, K_s and K_n—all play an important role in its operation and should be given suitable values. Furthermore, K, K_s, K_n can also be time varying, creating various "annealing schedules". As we show in Sect. 4, this feature gives us considerable flexibility in operating oscillator-based Ising machines for good performance.

3.4 Coupled oscillator networks with frequency variations

A major obstacle to the practical implementation of large-scale Ising machines is variability. While few analyses exist for assessing the effects of variability for previous Ising machine technologies (Sect. 2), the effect of variability on our oscillator-based Ising machine scheme is easy to analyse, predicting that performance degrades gracefully.

One very attractive feature of oscillators is that variability, regardless of the nature and number of elemental physical sources, eventually manifests itself essentially in only one parameter, namely the oscillator's natural frequency. As a result, the effect of variability in an oscillator network is that there is a spread in the natural frequencies of the oscillators. Incorporating frequency variability simply entails not making the assumption $\omega_i \equiv \omega^*$ that was used to derive (6) from (5). Without this assumption, (9) becomes

$$\frac{d}{dt}\phi_i(t) = \omega_i - \omega^* - \frac{\omega_i}{\omega^*} \cdot K \cdot \sum_{j=1,\, j\neq i}^{n} J_{ij} \cdot \sin(\phi_i(t) - \phi_j(t)) - \frac{\omega_i}{\omega^*} \cdot K_s \cdot \sin(2\phi_i(t)). \tag{15}$$

As it turns out, there is also a global Lyapunov function associated with this system:

$$E(\vec{\phi}(t)) = -\frac{K}{\omega^*} \cdot \sum_{i,j,\, i\neq j} J_{ij} \cdot \cos(\phi_i(t) - \phi_j(t)) - \frac{K_s}{\omega^*} \cdot \sum_{i=1}^{n} \cos(2\phi_i(t)) - 2\sum_{i=1}^{n} \frac{\omega_i - \omega^*}{\omega_i}\phi_i. \tag{16}$$

This can be proven as follows:

$$\begin{aligned} \frac{\partial E(\vec{\phi}(t))}{\partial \phi_k(t)} &= \frac{K}{\omega^*} \cdot \sum_{l=1,\, l\neq k}^{n} J_{kl} \cdot 2 \cdot \sin(\phi_k(t) - \phi_l(t)) \\ &\quad + \frac{K_s}{\omega^*} \cdot 2 \cdot \sin(2\phi_k(t)) - 2\frac{\omega_k - \omega^*}{\omega_k} \\ &= -\frac{2}{\omega_k} \cdot \frac{d\phi_k(t)}{dt}. \end{aligned} \tag{17}$$

Therefore,

$$\frac{dE(\vec{\phi}(t))}{dt} = -\sum_{k=1}^{n} \frac{2}{\omega_k} \left(\frac{d\phi_k(t)}{dt}\right)^2 \leq 0. \tag{18}$$

Note that (16) differs from (10) only by a weighted sum of ϕ_i—it represents essentially the same energy landscape, but tilted linearly with slopes proportional to the relative frequency variability. While it can still change the locations and values of the solutions, its effects are easy to analyse given a specific combinatorial optimisation problem. Also, as the coupling coefficient K becomes larger, the impact of variability is reduced. A small amount of variability merely perturbs the locations of minima a little, i.e., the overall performance of the Ising machine remains essentially unaffected. A very large amount of variability can, of course, eliminate minima that would exist if there were no variability. However, another great advantage of using oscillators is that even in the presence of large variability, the oscillator frequencies can be calibrated (e.g., using a voltage-controlled oscillator (VCO) scheme) prior to each run of the machine. As a result, the spread in frequencies can be essentially eliminated in a practical and easy-to-implement way.

4 Examples

In this section, we demonstrate the feasibility and efficacy of our oscillator-based Ising machine scheme by applying it to several MAX-CUT examples, a set of frustrated loop problems, and a graph colouring problem.

4.1 Small MAX-CUT problems

Given an undirected graph, the MAX-CUT problem (Myklebust 2015; Festa et al. 2002) asks us to find a subset of vertices such that the sum of the weights of the cut set between this subset and the remaining vertices is maximised. As an example, Fig. 4 shows a size-8 cubic graph, where each vertex is connected to three others—neighbours on both sides and the opposing vertex. As shown in Fig. 4, dividing the 8 vertices randomly yields a cut size of 5; grouping even and odd vertices, which one may think is the best strategy, results in a cut size of 8; the maximum cut is actually 10, with one of the solutions shown in the illustration. Changing the edge weights to non-unit values can change the maximum cut and also make the solution look less regular, often making the problem more difficult to solve. While the problem may not seem challenging at size 8, it quickly becomes intractable as the size of the graph grows. In fact, MAX-CUT is one of Karp's 21 NP-complete problems (Karp 1972).

The MAX-CUT problem has a direct mapping to the Ising model (Barahona 1982), by choosing J_{ij} to be the opposite of the weight of the edge between vertices i and j, i.e., $J_{ij} = -w_{ij}$. To explain this mapping scheme, we can divide the vertices into two sets—V_1 and V_2. Accordingly, all the edges in the graph are separated into three groups—those that connect vertices within V_1, those within V_2, and the cut set containing edges across V_1 and V_2. The sums of the weights in these three sets are denoted by S_1, S_2 and S_{cut}. Together, they constitute the total edge weights of the graph, which is also the negation of the sum of all the J_{ij}s:

$$S_1 + S_2 + S_{cut} = \sum_{i,j,\, i<j} w_{ij} = -\sum_{i,j,\, i<j} J_{ij}. \tag{19}$$

We then map this division of vertices to the values of Ising spins, assigning $+1$ to a spin i if vertex $v_i \in V_1$, and -1 if the vertex is in V_2. The Ising Hamiltonian in (2) can then be calculated as

$$
\begin{aligned}
H &= -\sum_{i,j,i<j} J_{ij}s_i s_j \\
&= -\sum_{i<j,\, v_i,v_j\in V_1} J_{ij}(+1)(+1) \\
&\quad -\sum_{i<j,\, v_i,v_j\in V_2} J_{ij}(-1)(-1) - \sum_{i<j,\, v_i\in V_1,\, v_j\in V_2} J_{ij}(+1)(-1) \\
&= -\sum_{i<j,\, v_i,v_j\in V_1} J_{ij} - \sum_{i<j,\, v_i,v_j\in V_2} J_{ij} + \sum_{i<j,\, v_i\in V_1,\, V_j\in V_2} J_{ij} \\
&= S_1 + S_2 - S_{cut} = \sum_{i,j,i<j} w_{ij} - 2\cdot S_{cut}.
\end{aligned}
\tag{20}
$$

Therefore, when the Ising Hamiltonian is minimised, the cut size is maximised.

To show that an oscillator-based Ising machine can indeed be used to solve MAX-CUT problems, we simulated the Kuramoto model in (9) while making the J_{ij}s represent the unit-weight cubic graph in Fig. 4. The magnitude of SYNC is fixed at $K_s = 3$, while we ramp up the coupling strength K from 0 to 5. Results from the deterministic model ($K_n = 0$) and the stochastic model ($K_n = 0.1$) are shown in Figs. 5 and 6 respectively. In the simulations, oscillators started with random phases between 0 and π; after a while, they all settled to one of the two phase-locked states separated by π. These two groups of oscillators represent the two subsets of vertices in the solution. The results for the 8 spins shown in Figs. 5 and 6 are $\{+1,-1,+1,-1,-1,+1,-1,+1\}$ and $\{-1,+1,+1,-1,+1,-1,-1,+1\}$ respectively; both are globally optimal solutions.

A minimal code for reproducing these results is shown in Wang and Roychowdhury (2019a). Note that these are simulations on stochastic differential equations with random initial conditions. Every run will return different waveforms; there is no guarantee that the global optimum will be reached on every run.

We have also directly simulated coupled oscillators at the SPICE level (using ngspice) to confirm the results obtained on phase macromodels. Such simulations are at a lower level than phase macromodels and are less efficient. But they are closer to physical reality and are useful for circuit design. In the simulations, 8 cross-coupled LC oscillators are tuned to a frequency of 5MHz. They are coupled through resistors, with conductances proportional to the coupling coefficients; in this case, we use $J_{ij}\cdot 1/100k\Omega$. Results from transient simulation using ngspice-28 are shown in Fig. 7. The 8 oscillators' phases settle into two groups {1,4,6,7} and {2,3,5,8}, representing one of the optimal solutions for the MAX-CUT problem. They synchronise within 20μs after oscillation starts, which is about 100 cycles. We have tried this computational experiment with different random initial conditions; like the phase macromodels, the SPICE-level simulations of these coupled oscillators reliably return optimal solutions for this size-8 MAX-CUT problem.

4.2 Hardware prototypes

We have built several OIM hardware prototypes (Wang et al. 2019), summarized in Fig. 8. They all use CMOS LC oscillators made with cross-coupled inverters (from TI SN74HC04N ICs), fixed inductors, trimmer capacitors and a 5V single supply. OIM8 and OIM32 (Wang and Roychowdhury 2019b) use 33μH inductors with capacitors tuned to around 30pF, for a natural frequency of 5MHz. In OIM8, resistors and potentiometers on the breadboard were plugged in manually and changed to program different problems, with results read out using oscilloscopes. For OIM32, rotary potentiometers were soldered on perfboards as couplings. Next to each potentiometer, we put male pin connectors to control the polarity of each connection, by shorting different pins using female jumper caps (color coded green and pink for positive and negative couplings). We soldered TI SN74HC86N Exclusive-OR (XOR) gate ICs to convert the oscillator phases to voltage levels, to power on-board LEDs for visualization and readout (via two 16-channel logic analyzers). We observed that for small-sized (8 and 32) Ising problems, global optima were achieved easily using these prototypes.

OIM64 and OIM240 use AD5206 digital potentiometer ICs (6-channel potentiometers with 8-bit accuracy) for programmability. Because these ICs are designed primarily for audio processing and do not have multi-MHz

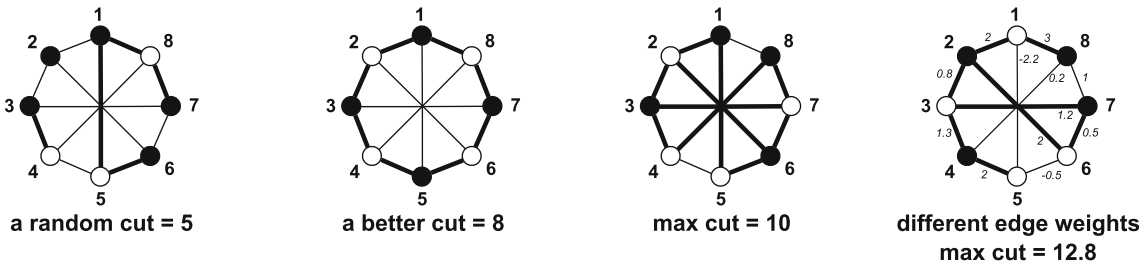


Fig. 4 Illustration of different cut sizes in a 8-vertex cubic graph with unit edge weights, and another one with random weights (rightmost)

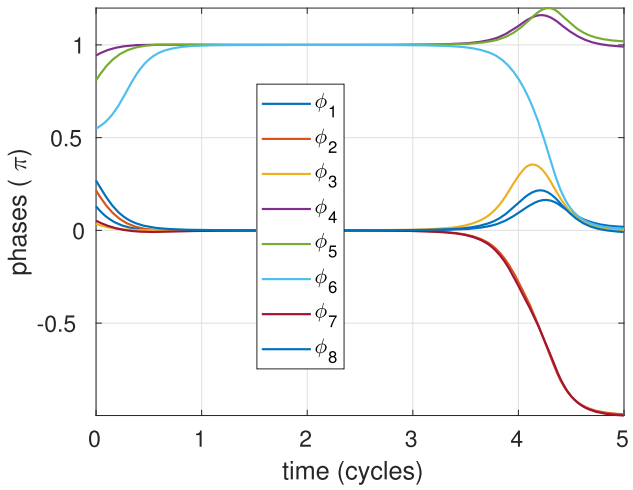


Fig. 5 Phases of oscillators solving a size-8 MAX-CUT problem without noise

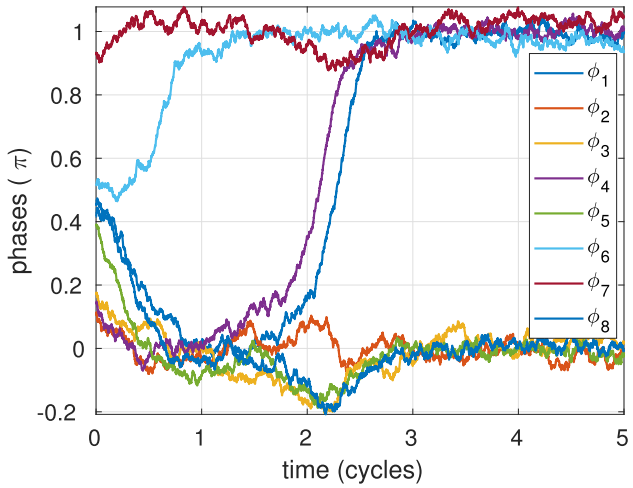


Fig. 6 Phases of oscillators solving a size-8 MAX-CUT problem with noise

bandwidth, we reduced oscillator frequencies to 1MHz. Both prototypes consist of multiple PCBs. OIM64 connects 64 oscillators in a 8x8 2D toroidal grid, with 192 couplings, each made of one channel of AD5206 and a SPDT switch for setting its polarity. Even though it was a bit cumbersome to "program" OIM64 due to the use of physical switches, we tried 10 randomly generated toroidal Ising grid instances, achieving the global optimum for each one.

In OIM240, we improved the design to use the position of the potentiometer wiper to switch polarity, thus eliminating the use of switches and making the coupling software programmable. On each PCB, we implemented 12 oscillators with a denser connectivity; 20 such PCBs were plugged into a motherboard through edge connectors, and interconnected in a 4×5 toroidal grid, implementing a total of 240 oscillators with 1200 couplings. The motherboard also distributes CLK, data lines and address lines for programming the 200 AD5206 ICs and for reading oscillator states, all controlled by an Arduino module on the motherboard that communicates with a PC through USB. When operating OIM240, we flip on the supply digitally, wait 1ms for oscillators to synchronize, then read back the solution. Even with all the overhead from serial reading, solutions can be read back every 3.5 ms. OIM240's operation consumes $\sim$5W of power for all the oscillators and peripheral circuitry, excluding only the LEDs.

We tested OIM240 with many randomly generated Ising problems (with each of the 1200 couplings randomly chosen from 0, -1, $+1$). A typical histogram for the energy levels of the measured solutions is shown in Fig. 8c. Note that a random (trivial) solution has an energy around 0, whereas the best polynomial-time algorithm (based on SDP) guarantees that 87.8% of the global optimum will be achieved. In comparison, results from OIM240 center around a very low energy, and achieve the global optimum multiple times. We performed the same measurements for 20 different random Ising problems, with the distances of solutions from their respective global optima[7] shown in Fig. 8d. The fact that OIM240 is finding highly non-trivial solutions indicates that it indeed physically implements a working and effective Ising machine.

4.3 MAX-CUT benchmark problems

In this section, we demonstrate oscillator-based Ising machines on larger-scale MAX-CUT problems. Specifically, we have run simulations on all the problems in a widely used set of MAX-CUT benchmarks known as the G-set [46,47].[8] In Table 1, in the column labelled OIM, we list the results of our OIM simulations alongside those from several heuristic algorithms developed for MAX-CUT—Scatter Search (SS) (Martí et al. 2009), CirCut (Burer et al. 2002), and Variable Neighbourhood Search with Path Relinking (VNSPR) (Festa et al. 2002).[9] We also list the performances of simulated annealing from a recent study (Myklebust 2015), the only one we were able to find that contains results for all the G-set problems.

For each problem, we ran 200 simulations of an OIM; each simulation had a randomly generated initial condition and injected random noise. The maximum cut value found over these 200 simulations is reported in Table 1. The column labelled $n_{\max}$ counts the number of simulations in which the OIM generated a cut with this maximum value.

[7] We ran simulated annealing for a long time (1m) multiple times, then treated the best results as global optima.

[8] Benchmarks G1 $\sim$ 21 are of size 800; G22 $\sim$ 42 are of size 2000; G43 $\sim$ 47, G51 $\sim$ 54 are of size 1000; G48 $\sim$ 50 are of size 3000.

[9] Results and runtimes for SS, CirCut and VNSPR are available in the "Computational Experiences" section of Wang (2017).

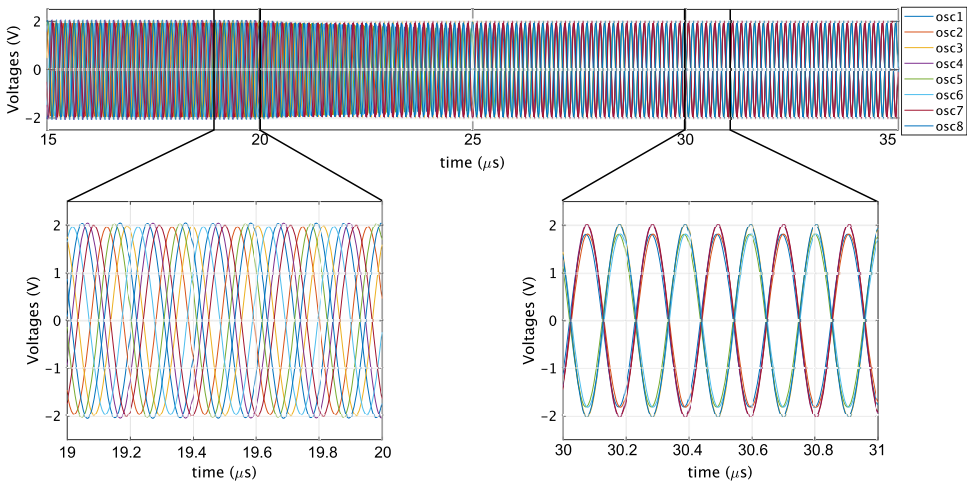
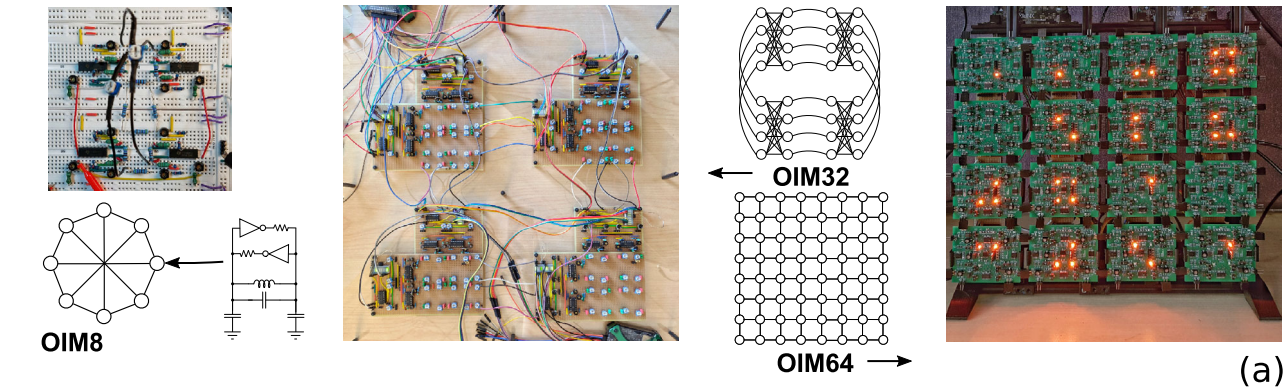


Fig. 7 Circuit-level simulation results: ngspice on 8 coupled oscillators



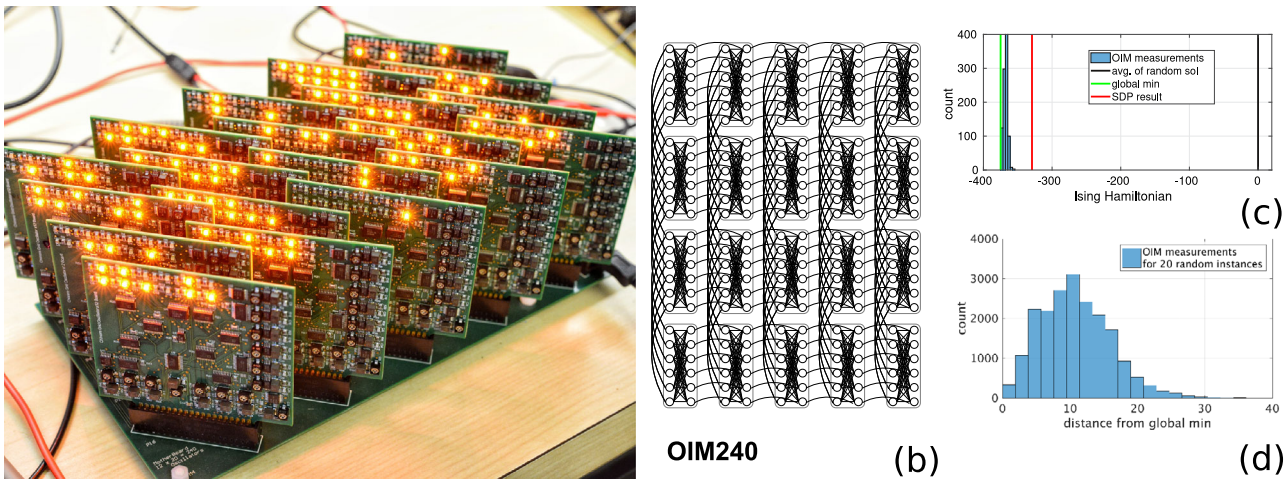


Fig. 8 OIM prototypes: (**a**) photos and schematics of OIM8, OIM32, OIM64; (**b**) OIM240; (**c**) energy levels of 1000 measured solutions from OIM240, on a random instance of size-240 Ising problem; (**d**) energy levels for 20 such random instances (showing the distances to their respective global minimum)

Similarly, the column labelled $n_{0.999}$ counts the number of simulations which generated cuts that were within 99.9% of the value reported in the OIM column.

The OIM Time column of Table 1 reports the total time it took for all 200 simulations to run on a single processor core.[10] While we list runtime results for each algorithm in

[10] Note that the 200 simulations can be run in parallel to greatly reduce this runtime. However, we stress that software simulation time is not of immediate relevance for OIM; instead, it is the time the OIM hardware will take to solve the problem, as discussed below.

Table 1 Overview of OIM run on MAX-CUT benchmarks in the G-set, compared with several heuristic algorithms

Benchmark	(|V|, |E|)	SS	Time	CirCut	Time	VNSPR	Time	SA	Time	OIM	Time	$n_{\max}$	$n_{0.999}$
G1	(800, 19,176)	**11,624**	139	**11,624**	352	11,621	22,732	11,621	295	**11,624**	10,520	14	57
G2	(800, 19,176)	**11,620**	167	11,617	283	11,615	22,719	11,612	327	11,619	10,540	1	39
G3	(800, 19,176)	**11,622**	180	**11,622**	330	**11,622**	23,890	11,618	295	11,621	10,480	2	50
G4	(800, 19,176)	**11,646**	194	11,641	524	11,600	24,050	11,644	294	**11,646**	10,540	7	44
G5	(800, 19,176)	**11,631**	205	11,627	1128	11,598	23,134	11,628	300	11,630	10,520	4	46
G6	(800, 19,176)	2165	176	**2178**	947	2102	18,215	**2178**	247	2177	10,560	4	9
G7	(800, 19,176)	1982	176	2003	867	1906	17,716	**2006**	205	**2006**	10,580	1	1
G8	(800, 19,176)	1986	195	2003	931	1908	19,334	**2005**	206	2004	10,560	2	11
G9	(800, 19,176)	2040	158	2048	943	1998	15,225	**2054**	206	2051	10,520	1	6
G10	(800, 19,176)	1993	210	1994	881	1910	16,269	1999	205	**2000**	10,580	1	3
G11	(800, 1600)	562	172	560	74	**564**	10,084	**564**	189	**564**	1340	69	69
G12	(800, 1600)	552	242	552	58	**556**	10,852	554	189	**556**	1260	57	57
G13	(800, 1600)	578	228	574	62	580	10,749	580	195	**582**	1280	39	39
G14	(800, 4694)	3060	187	3058	128	3055	16,734	**3063**	252	3061	2920	3	32
G15	(800, 4661)	3049	143	3049	155	3043	17,184	3049	220	**3050**	3220	3	47
G16	(800, 4672)	3045	162	3045	142	3043	16,562	3050	219	**3051**	2900	3	13
G17	(800, 4667)	3043	313	3037	366	3030	18,555	**3045**	219	3044	2920	1	26
G18	(800, 4694)	988	174	978	497	916	12,578	990	235	**992**	2940	1	1
G19	(800, 4661)	903	128	888	507	836	14,546	904	196	**906**	2900	5	5
G20	(800, 4672)	**941**	191	**941**	503	900	13,326	**941**	195	**941**	2940	47	47
G21	(800, 4667)	930	233	**931**	524	902	12,885	927	195	930	2920	6	6
G22	(2000, 19,990)	13,346	1336	13,346	493	13,295	197,654	13,158	295	**13,354**	11,740	1	22
G23	(2000, 19,990)	13,317	1022	13,317	457	13,290	193,707	13,116	288	**13,330**	11,720	1	21
G24	(2000, 19,990)	13,303	1191	13,314	521	13,276	195,749	13,125	289	**13,321**	11,800	1	18
G25	(2000, 19,990)	13,320	1299	**13,326**	1600	12,298	212,563	13,119	316	13,322	11,740	1	26
G26	(2000, 19,990)	13,294	1415	**13,314**	1569	12,290	228,969	13,098	289	13,312	11,780	1	16
G27	(2000, 19,990)	3318	1438	3306	1456	3296	35,652	**3341**	214	3320	11,800	4	6
G28	(2000, 19,990)	3285	1314	3260	1543	3220	38,655	**3298**	252	3290	12,240	4	6
G29	(2000, 19,990)	3389	1266	3376	1512	3303	33,695	3394	214	**3400**	11,780	1	1
G30	(2000, 19,990)	3403	1196	3385	1463	3320	34,458	**3412**	215	3408	11,800	1	2
G31	(2000, 19,990)	3288	1336	3285	1448	3202	36,658	**3309**	214	3294	11,820	1	5
G32	(2000, 4000)	1398	901	1390	221	1396	82,345	**1410**	194	1408	3500	2	2
G33	(2000, 4000)	1362	926	1360	198	1376	76,282	1376	194	**1380**	3180	3	3
G34	(2000, 4000)	1364	950	1368	237	1372	79,406	1382	194	**1384**	3180	1	1
G35	(2000, 11,778)	7668	1258	7670	440	7635	167,221	7485	263	**7676**	7420	1	6
G36	(2000, 11,766)	7660	1392	7660	400	7632	167,203	7473	265	**7663**	7520	2	16
G37	(2000, 11,785)	7664	1387	7666	382	7643	170,786	7484	288	**7674**	7560	1	20
G38	(2000, 11,779)	**7681**	1012	7646	1189	7602	178,570	7479	264	7674	7540	1	7
G39	(2000, 11,778)	2393	1311	2395	852	2303	42,584	**2405**	209	2397	7440	1	2
G40	(2000, 11,766)	2374	1166	**2387**	901	2302	39,549	2378	208	**2387**	7620	1	1
G41	(2000, 11,785)	2386	1017	2398	942	2298	40,025	**2405**	208	2401	7560	3	9
G42	(2000, 11,779)	2457	1458	2469	875	2390	41,255	2465	210	**2470**	7460	1	1
G43	(1000, 9990)	6656	406	6656	213	6659	35,324	6658	245	**6660**	5820	1	46
G44	(1000, 9990)	**6648**	356	6643	192	6642	34,519	6646	241	6647	5840	3	55
G45	(1000, 9990)	6642	354	6652	210	6646	34,179	6652	241	**6653**	5820	7	13
G46	(1000, 9990)	6634	498	6645	639	6630	38,854	**6647**	245	6645	5820	2	24
G47	(1000, 9990)	6649	359	**6656**	633	6640	36,587	6652	242	**6656**	5820	4	18
G48	(3000, 6000)	**6000**	20	**6000**	119	**6000**	64,713	**6000**	210	**6000**	4640	193	193

Table 1 (continued)

Benchmark	(|V|, |E|)	SS	Time	CirCut	Time	VNSPR	Time	SA	Time	OIM	Time	$n_{\max}$	$n_{0.999}$
G49	(3000, 6000)	**6000**	35	**6000**	134	**6000**	64,749	**6000**	210	**6000**	4640	127	127
G50	(3000, 6000)	**5880**	27	**5880**	231	**5880**	147,132	5858	211	5876	5120	3	45
G51	(1000, 5909)	**3846**	513	3837	497	3808	89,966	3841	234	3845	3680	3	20
G52	(1000, 5916)	**3849**	551	3833	507	3816	95,985	3845	228	3848	3680	1	6
G53	(1000, 5914)	**3846**	424	3842	503	3802	92,459	3845	230	3845	3680	3	30
G54	(1000, 5916)	3846	429	3842	524	3820	98,458	3845	228	**3850**	3700	6	12

Bold entries indicate highest cut values across all columns. Runtimes are in seconds. The OIM Time column reports the total CPU time of our 200 simulations of the SDE. Notably, this workload is highly parallelizable. As described elsewhere in this section, on a physical OIM, the running time of 200 optimizations would be on the order of milliseconds, significantly faster than running algorithms in software

Table 1, note that they come from different sources and were measured on different platforms. Results for SS, CirCut and VNSPR were obtained from Dual Intel Xeon at 3.06 GHz with 3.2 GB of RAM; SA was run on Intel Xeon E3-1245v2 at 3.4 GHz with 32 GB of RAM (Myklebust 2015). To make the results generally comparable, we ran our simulations on a modest personal desktop with Intel Xeon E5-1603v3 at 2.8 GHz with 16 GB of RAM. Another notable feature of our method is that unlike other algorithms, SDE simulation does not know about the Ising Hamiltonian or cut value—it never needs to evaluate the energy function or relative energy changes, which are implicit in the dynamics of differential equations.

The data in Table 1 shows that our oscillator-based Ising machine is indeed effective—it finds best-known cut values for 29 out of the 54 problems, 17 of which are better than those reported in the above literature.[11] Moreover, in the 200 random trials, the best cut is often reached more than once—the average $n_{\max}$ for all benchmarks is 20 out of 200. Further, the relaxed objective of $n_{0.999}$ has an average of 56, i.e., very good cuts are found in more than a quarter of the total trials.

The results can be improved further if we tailor the "annealing schedule" for each problem. For simplicity, and also to evaluate how necessary it is to tailor schedules to obtain good results, we used a single annealing schedule for all the problems. This schedule was chosen empirically for one problem, G1, and appears to work well for most G-set problems, as is apparent from the table. Rather than using constants for K, K_s, and K_n from (14), we increased the coupling strength K linearly with time, linearly increased the noise level K_n from 0 to 1 early in the simulation, and ramped the SYNC's amplitude K_s up and down multiple times. Figure 9 shows how K, K_s, and K_n vary with time and also plots the oscillator phases and the instantaneous cut values when solving the G1 benchmark to its best-known cut size. MATLAB® code illustrating the "annealing schedule" is shown in Wang and Roychowdhury (2019a); we used a much faster implementation in C++ to generate the results in Table 1. In time, we plan to release our code as open-source software.

The fact that we were using the same schedule for all the benchmarks implies that the actual hardware time for the Ising machine to solve all these benchmarks would be the same, regardless of problem size and connectivity. Note that in Fig. 9, the end time 20 means 20 oscillation cycles, but this end time is predicated on a coupling strength of $K \sim 1$. The actual value of K for each oscillator depends on the nature of the oscillators (in particular their PPV functions and nominal oscillation waveform shapes), as we show in the derivation of Gen-Adler in Wang and Roychowdhury (2019a). As an example, for the LC oscillators we use in Sect. 4.1 with 100k resistive coupling, $K \approx 0.02$. This indicates that it takes less than 100 cycles for the oscillators to synchronise in phase, which is consistent with measurements. For such a coupled LC oscillator network, a hardware time of 20 in Fig. 9 represents approximately 2000 cycles of oscillation; for 5 MHz oscillators, this takes 0.4 ms. If we use GHz nano-oscillators, the computation time can be well within a microsecond. As comparison, the runtime of the several heuristic algorithms listed in Table 1, even with faster CPUs and parallel implementations in the future, is unlikely to ever drop to this range.

We also ran more computational experiments on the G-set benchmarks in order to gauge the effects of adding noise, SYNC ramping, the nature of the c_{ij} function in (5), and frequency variability. For each experiment, we re-ran each of the 54 benchmarks 200 times and recorded the best cut value from each run. In Fig. 10, we compare the quality of these cut values with results from the baseline (unaltered) runs by plotting histograms (over all 10,800 runs for the entire G-set) of the distances of the cut values to their respective maxima.

[11] The results here were re-generated using updated code, hence do not exactly match the results we reported previously (Wang and Roychowdhury 2019a, b).

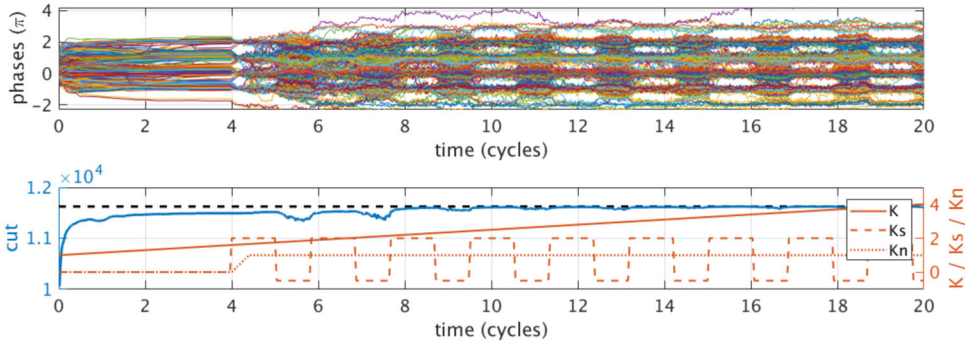


Fig. 9 Coupled oscillators solving MAX-CUT benchmark problem G1 (Helmberg and Rendl 2000) to its best-known cut size 11,624

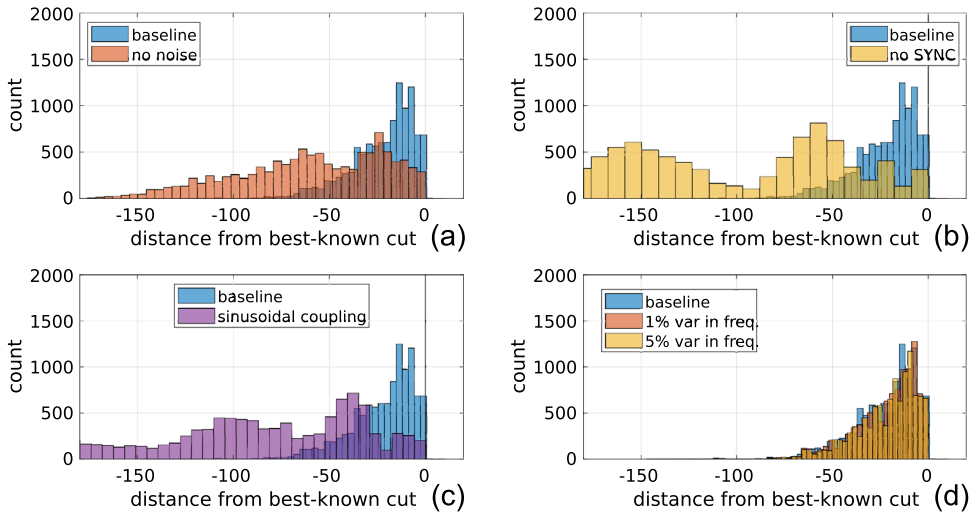


Fig. 10 Histograms of cut values for different modifications of the Ising machine/simulation, compared with the baseline used for generating in Table 1. In (**a**), noise has been removed from the simulation ($K_n = 0$). The SYNC signal has been removed in (**b**). $c_{ij}(\cdot)$ (in (5)) has been changed in (**c**), and frequency variability has been added in (**d**). Results indicate that noise, SYNC and appropriately choosing $c_{ij}(\cdot)$ functions are key to OIM's operation, whereas moderate frequency variability does not affect OIM's performance much

In the first experiment, we removed noise from the simulation by setting $K_n \equiv 0$. The solutions become considerably worse as seen in Fig. 10a, confirming that noise helps the coupled oscillator system settle to lower energy states. In the next experiment (Fig. 10b), we eliminated SYNC entirely by setting $K_s \equiv 0$. Without SYNC, the system becomes a simple coupled oscillator system with phases that take a continuum of values, as discussed in Sect. 3.2. The settled analog values of the phases that were then thresholded to 0 or π to correspond to Ising spins. As shown in Fig. 10, the results become much worse; indeed, none of the best-known results were reached. This indicates that the SYNC signal and the mechanism of SHIL we introduce to the coupled oscillator networks are indeed essential for them to operate as Ising machines.

Our baseline Ising machine actually uses a smoothed square function $\tanh(\sin(\cdot))$ for the $c_{ij}(\cdot)$ functions in (5), as opposed to the $\sin(\cdot)$ used in the original Kuramoto model, as shown in the code in Wang and Roychowdhury (2019a). This changes the $\cos(\cdot)$ term in the energy function (10) to a triangle function. Such a change appears to give better results than the original, as shown in Fig. 10c. The change requires designing oscillators with special PPV shapes and waveforms such that their cross-correlation is a square wave, which is not difficult in practice based on our derivation in Wang and Roychowdhury (2019a). As an example, rotary travelling wave oscillators naturally have square PPVs. Ring oscillators can also be designed with various PPVs and waveforms by sizing each stage individually. We cannot say definitively that the square function we have used is optimal for Ising solution performance, but the significant improvement over sinusoidal coupling functions indicates that a fruitful direction for further exploration may be to look beyond the original Kuramoto model for oscillator-based computing.

The last experiment we report here (Fig. 10d) adds variability to the natural frequencies of the oscillators, as in (15). We assigned Gaussian random variables to the ω_is,

with ω^* as the mean, and 0.01 (1%) and 0.05 (5%), respectively, as the standard deviations for two separate runs. From Fig. 10d, we observe that even with such non-trivial spread in the natural frequencies of oscillators, the performance is affected very little.

Finally, we conducted a preliminary study of the scaling of the time taken by the Ising machine to reach good solutions as problem sizes increase. As the G-set benchmarks have only a few sizes (800, 1000, 2000 and 3000), we used the program (named `rudy` [50]) that generated them to create more problems of various sizes. All generated problems used random graphs with 10% connectivity and ± 1 coupling coefficients. We simulated all of them, each for 200 instances, with fixed parameters $K = 1$, $K_s = 0.1$, $K_n = 0.01$, and show all their Ising Hamiltonians over time in Fig. 11. Much to our surprise, the speed in which the values settle appears almost constant, regardless of the problem size. While this does not necessarily mean they all converge to the global optima within the same time, this preliminary study is encouraging as it confirms the massively parallel nature of the system. For larger Ising problems, our Ising machine only needs to scale linearly in hardware size with the number of spins, but does not necessarily require much more time to reach a solution.

4.4 Frustrated loop problems

To best explore runtimes of any statistical algorithm or probabilistic machine for solving combinatorial optimisation problems, it is helpful to have benchmark problems that (1) can be of any desired size, and (2) feature *known global minima* of the objective function. Such problem sets make it possible to assess, quantitatively, how well the algorithm/machine is finding the known global minimum, and how long it is taking to do it. The G-set MAX-CUT benchmarks in Sect. 4.3, although widely used, are not suitable for this purpose. They consist of problems of only limited sizes (800, 2000, 3000) and do not have provable globally optimal cuts—as a result, only best-known results can be used for benchmarking and comparison. However, a special set of benchmarks, known as "frustrated loop" problems, has been proposed (Hen et al. 2015; King et al. 2015; Sheldon et al. 2019) to address this issue. In this section, we present a preliminary exploration of how OIM's runtime scales with problem size on frustrated loop problems.

4.4.1 Frustrated loops

Consider a special Ising graph, i.e., one that forms a simple loop. Frustrated loop Ising problems are constructed by sharing edges between many such simple loops, to form more complex graphs. If loops are constructed, and edges shared, in a particular way, it can be shown that the Ising Hamiltonian of the final graph is the sum of those of the separate loops (Hen et al. 2015). By creating loops with "planted" solutions that have known minimum Hamiltonians, the global minimum Hamiltonian and solution for the final graph also become known.

To understand how a solution can be planted, consider a loop made of L spins and L edges. Predefine a solution $\vec{s}^*$, with spin values $s_i^* \in \{+1, -1\}, i = 1, 2, \ldots, L$. If the L edge weights (J_{ij} terms) are set to be

$$J_{ij} = s_i^* \cdot s_j^*, \tag{21}$$

the Ising Hamiltonian at $\vec{s}^*$ becomes

$$H(\vec{s}^*) = \sum_{i<j} -J_{ij} s_i^* s_j^* = \sum_i -1 = -L. \tag{22}$$

This is obviously the global minimum for this single loop problem, as every term in the summation is minimized.

The loop becomes "frustrated" if some of the J_{ij}s are made different from (21)—more generally, if for *any choice of* planted solution, not all J_{ij}s can obey (21), i.e., not all L terms can reach -1 at the same time, then the loop is called frustrated. A special case of such frustrated loops is when only one of the L edge weights is flipped in sign from -1 to $+1$—i.e., a loop with one frustrated edge. It has been shown that in this case, the global optimal Hamiltonian becomes $-L + 2$, with the pre-defined solution $\vec{s}^*$ being one of the global optimal solutions (Hen et al. 2015).

From a set of N spins with a planted solution imposed, it is possible to take M such frustrated loops and "connect" them together. If several loops are merged at an edge, the weight of the merged edge becomes simply the sum of the original weights of the edges. Given that all the individual frustrated loops were constructed with the same planted solution $s_i^* \in \{+1, -1\}$, $i = 1, 2, \ldots, N$, and that there is one frustrated edge per loop,[12] it can be shown that the global Hamiltonian minimum is $2M - \sum_{i=1}^{M} L_i$, where L_i is the length of the i^{th} loop (Sheldon et al. 2019).

Frustrated loop problems are typically generated on an underlying lattice. The lattice can be D-Wave's Chimera graph (Hen et al. 2015; King et al. 2015), or 2-D/3-D grids with nearest neighbour connections (Sheldon et al. 2019). That is to say, the loops are formed from existing edges in these lattices. As an illustration, Fig. 12 shows two loops in a small 3-D grid, with blue and red edges representing positive and negative J_{ij} terms respectively. While this particular problem is easily solvable by considering the two loops separately, such problems become much harder with

[12] Each single loop has one frustrated edge by design. When they overlap, the weights of the frustrated edges can change, and they may not remain individually frustrated. This does not change the result regarding the global minimum energy level.

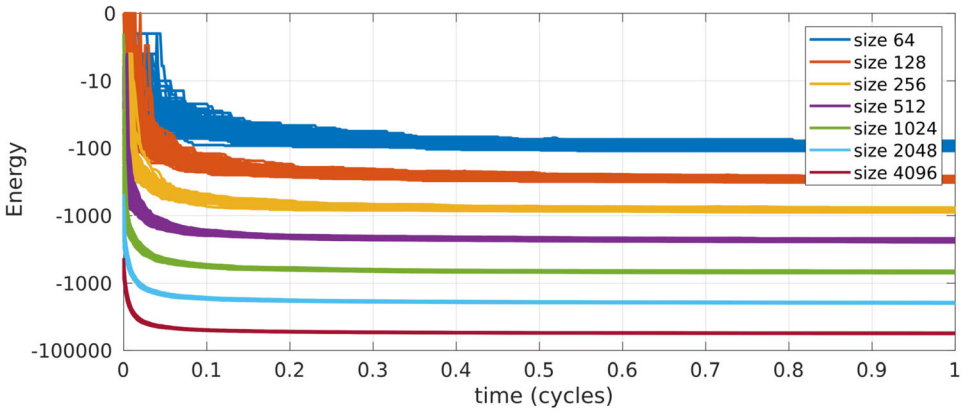


Fig. 11 Speed of energy minimisation for problems of different sizes

denser loops on larger grids. Figure 13 displays a problem with 2400 loops on an underlying 20-by-20-by-20 3-D grid. As it happens, when the loops overlap heavily, there is no known algorithm that can efficiently recover the loops from the graph (hence compute the global minimum) (Sheldon et al. 2019). During construction, a parameter α is used to control the density of loops—it is the ratio between the number of loops and the number of vertices, i.e., $M = \alpha N$. When using 3-D grids as the underlying lattice, explorations have indicated that problems are hardest when α is around 0.3.

With the goal of reproducing the types of frustrated loop problems used in Sheldon et al. (2019), the above discussion can be summarized into a procedure for generating frustrated loops:

- Start with an N_L-by-N_L-by-N_L toroidal 3-D grid with a number $N = N_L^3$ of vertices—$\{v_{p,q,k}\}$, where $p, q, k = 1, 2, \ldots, N_L$. In such a 3-D grid, each vertex $v_{p,q,k}$ has edge connections to six "neighbours"—the vertices on its "left", "right", "up", "down", "front" and "back" directions. We call the grid toroidal as the vertices on its boundaries connect to those on the opposite sides. For example, the "left neighbour" of $v_{p,q,k}$ is defined as: $v_{p-1,q,k}$ when $p > 1$, or $v_{N_L,q,k}$ when $p = 1$—there is an edge connecting the left-most vertex with its corresponding right-most one. Similar toroidal connections exist along the other directions as well.

  We treat the N vertices as N spins, and initialize all edge weights J_{ij} to zero.
- Predefine or "plant" a set of solutions $\{s_i^*\}$, where $i = 1, 2, \ldots, N$, with each s_i^* randomly chosen to be $+1$ or -1.
- To generate a loop, start from a random vertex and perform a random walk (randomly taking one of the six possible directions at each step) until the path crosses itself, then keep the loop and discard the vertices not in the loop.
- Check if the loop is too short.[13] If so, discard the whole loop. If not, randomly choose an edge to be the frustrated edge[14] and assign its J_{ij} as $-s_i^* \cdot s_j^*$; assign all other J_{ij} edge weights to be $+s_i^* \cdot s_j^*$. Then add these edge weights to the total J matrix of the graph.
- Repeat the generation of loops until the number of valid (not too short) loops reach $\alpha \cdot N$.

Since we know the length of each loop and the number of loops, for every instance generated, we also know the lowest achievable Hamiltonian value and the "planted" globally optimal solution. With a randomly generated set of such frustrated loop problems, we can explore how effective a given algorithm is for finding (known) global optima.

When running a statistical algorithm on a given problem, reporting the time it takes for one run to reach the global optimum is often not enough. We are often more interested in knowing the total runtime needed for the algorithm to achieve the global optimum more or less reliably. Such a concept can be concretized as the *time to solution*, or *TTS* (Hamerly et al. 2019), given by

$$TTS = \frac{\log(1-P)}{\log(1-p)} T, \tag{23}$$

where T is the length of a simulation run, $P = 0.99$ is the (desired) probability of achieving a global minimum (over a simulation of duration TTS), and p is the probability with

[13] Different criteria are used in different versions of frustrated loop benchmarks. In Sheldon et al. (2019), a loop is too short when its length is smaller than 6; in Hen et al. (2015), when its length is smaller than 8; in King et al. (2015), when the loop remains confined in a 8-vertex cube. We choose to be consistent with Sheldon et al. (2019) for this preliminary exploration.

[14] In Sheldon et al. (2019) and Hen et al. (2015), the frustrated edge is chosen randomly; in King et al. (2015), the frustrated edge is chosen randomly among edges with a total weight below a certain threshold, such that ideally most edge weights reach the threshold eventually.

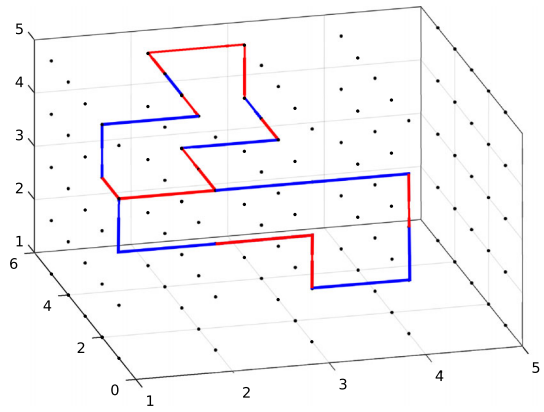


Fig. 12 Two frustrated loops in a 3-D lattice with one overlapping edge. Blue and red edges represent positive and negative J_{ij} terms respectively (Color figure online)

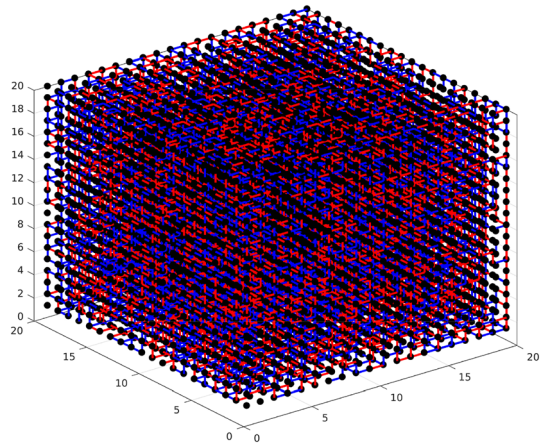


Fig. 13 A size-8000 frustrated loop instance on a 20-by-20-by-20 3-D lattice

which a simulation of duration T achieves the global minimum.

Note that (23) estimates the TTS given a runtime T, and a corresponding probability p of finding the global minimum in this runtime. In practice, using different T values can lead to different TTS values, since successive runs are typically not statistically independent for nonlinear dynamical systems, especially for small T, even if random initial conditions are used and statistically independent input noise is applied. Therefore, in practice, TTS is defined to be the one for the **best-case scenario**. That is to say, it is assumed that we can find an optimal set of parameters (including runtime T) such that the TTS calculated in (23) is minimized. Needless to say, this is hard to ensure in practice. The results reported (Sheldon et al. 2019; Hamerly et al. 2019) are always the upper bound of this "true TTS" of the algorithms.

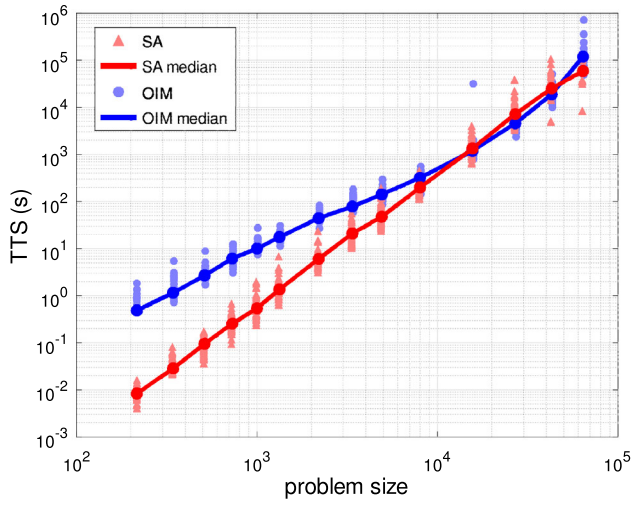


Fig. 14 TTS from SA and OIM simulations of 3-D grid frustrated loop instances

4.4.2 Experimental set-up and results

Our first goal here is to reproduce the benchmarks used in Sheldon et al. (2019) statistically, i.e., to generate random frustrated loop instances on 3-D grids of multiple sizes, following the benchmark description in Sheldon et al. (2019). The lengths of the grids are {6, 7, 8, 9, 10, 11, 13, 15, 17, 20, 25, 30, 35, 40}, corresponding to spin numbers ranging from 216 to 64,000. One peculiar observation from Sheldon et al. (2019) is that the TTS of simulated annealing scales exponentially with problem size, whereas the TTS of a heuristic algorithm called "memcomputing" scales polynomially. Here we report our own observations on the TTS of simulated annealing (SA), and also preliminary[15] results for OIM, running on these frustrated loop problems.

We generated 50 instances with different random seeds for each problem size. For each instance, we ran SA and OIM with various values of T, performing 100 runs for each T to measure success probability. Finally, we took the minimum TTS over all values of T; these are plotted in Fig. 14. From the scatter plots in Fig. 14, we observe that for any given problem size, the 50 instances used in this experiment have a spread of TTS values, indicating a varying degree of computational difficulty. The median of the TTSs for each problem size, depicted with darker lines in Fig. 14, is also shown, to indicate the trend of TTS scaling.

From Fig. 14, we observe that the TTS of SA (our own simple implementation, from first principles) does not scale exponentially on these frustrated loop problems; instead, the almost-straight line on the log-log plot indicates

[15] e.g., we have not yet explored OIM parameter tuning to achieve lower TTS.

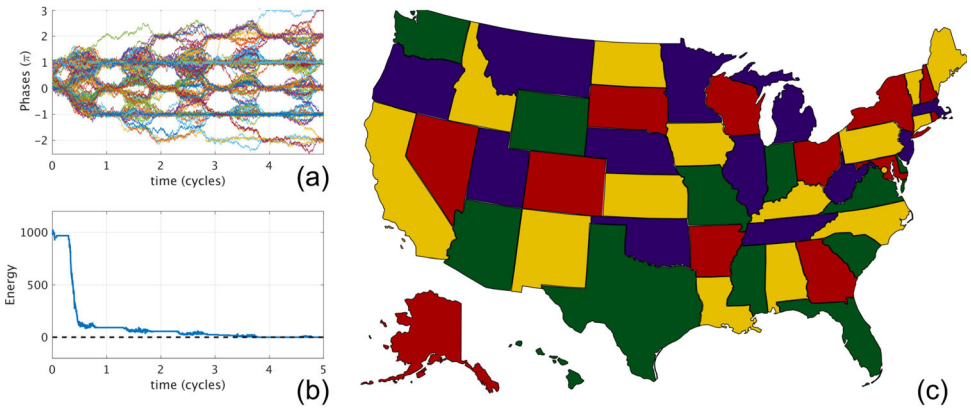


Fig. 15 Coupled oscillators colouring the states in the US map: (**a**) phases of oscillators evolve over time; (**b**) energy function (24) decreases during the process; (**c**) the resulting US map colouring scheme

polynomial scaling. Similarly, the TTS of OIM also scales roughly polynomially—note that further experiments, after parameter tuning, may yield better results. The TTS values of OIM simulation and SA are similar for large-sized problems.

4.5 A graph colouring example

As mentioned in Sect. 2, many problems other than MAX-CUT can be mapped to the Ising model (Lucas 2013) and solved by OIM. Here we show an example of a graph colouring problem—assigning four colours to the 51 states (including a federal district) of the United States such that no two adjacent states have the same colour.[16]

Each state is represented as a vertex in the graph. When two states are adjacent, there is an edge in the graph that connects the corresponding vertices. For every vertex i, we assign four spins s_{iR}, s_{iG}, s_{iB} and s_{iY} to represent its colouring scheme; when only one of them is $+1$, the vertex is successfully coloured as either red, green, blue or yellow. Then we write an energy function H associated with these $4 \times 51 = 204$ spins as follows:

$$\begin{aligned} H = &\sum_{i}^{n}(2 + s_{iR} + s_{iG} + s_{iB} + s_{iY})^2 \\ &+ \sum_{(i,j)\in\mathbb{E}}^{n_{\mathbb{E}}} \big[(1 + s_{iR})(1 + s_{jR}) + (1 + s_{iG})(1 + s_{jG}) \\ &+ (1 + s_{iB})(1 + s_{jB}) + (1 + s_{iY})(1 + s_{jY})\big], \end{aligned} \tag{24}$$

where $n = 51$ is the number of vertices, $\mathbb{E}$ represents the edge set, and $n_{\mathbb{E}}$ is the number of edges—in this case equal to 220.[17]

The first term of H is a sum of squares never less than zero; it reaches zero only when $\{s_{iR}, s_{iG}, s_{iB}, s_{iY}\}$ contains three -1s and one $+1$ for every i, i.e., each state has a unique colour. The latter term is also a sum that is always greater than or equal to zero, as each spin can only take a value in $\{-1, +1\}$; it is zero when $s_{iX} = s_{jX} = +1$ never occurs for any edge connecting i and j, and for any colour $X \in \{R, G, B, Y\}$, i.e., adjacent states do not share the same colour. Therefore, when H reaches its minimum value 0, the spin configuration represents a valid colouring scheme—following the indices of the $+1$ spins $\{i, X \mid s_{iX} = +1\}$, we can then assign colour X to state i.

Note that when expanding the sum of squares in (24), we can use the fact that $s_{iX}^2 \equiv 1$ to eliminate the square terms. This means H contains only products of two spins—modelled by J_{ij}s, and self terms—modelled by h_i. These Ising coefficients can then be used to determine the couplings in an oscillator-based Ising machine.

We simulated these 204 coupled oscillators; the results are shown in Fig. 15. In the simulation, we kept K and K_n constant while ramping K_s up and down 5 times. We found the Ising machine to be effective with this schedule as it could colour the map successfully in more than 50% of the random trials and returned many different valid colouring schemes.

5 Conclusion

In this paper, we have proposed a novel scheme for implementing Ising machines using self-sustaining nonlinear oscillators. We have shown how coupled oscillators naturally minimise an "energy" represented by their global Lyapunov function, and how introducing the mechanism of subharmonic injection locking modifies this function to encode the Ising Hamiltonian. The validity and feasibility of the scheme have been examined via phase-model and circuit-level simulations, as well as proof-of-concept hardware implementations. Simulations of larger-scale

[16] Ising machines can be used on general graph colouring problems, and this four-colouring problem is chosen here for illustrative purposes. Four-colouring a planar graph is actually not NP-hard and there exist polynomial-time algorithms for it Robertson et al. (1996).

[17] Hawaii and Alaska are considered adjacent, hence their colours will be different in the map.

benchmark problems have also shown promising results in both speed and the quality of solutions. We believe that our scheme constitutes an important and practical means for the implementation of scalable Ising machines.

Acknowledgements The authors are indebted to the anonymous reviewers for their exceptionally well-informed and thorough examination of the manuscript and their detailed, insightful suggestions for improvement. Support from the US National Science Foundation is gratefully acknowledged.

References

Acebrón JA, Bonilla LL, Vicente CJP, Ritort F, Spigler R (2005) The Kuramoto model: a simple paradigm for synchronization phenomena. Rev Mod Phys 77(1):137

Aramon M et al (2019) Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. Front Phys 7:48

Barahona Francisco (1982) On the computational complexity of Ising spin glass models. J Phys A Math Gen 15(10):3241

Bhansali P, Roychowdhury J (2009) Gen-Adler: the generalized Adler's equation for injection locking analysis in oscillators. In: Proceedings of IEEE ASP-DAC, pp 227–522

Bian Z, Chudak F, Israel R, Lackey B, Macready WG, Roy A (2014) Discrete optimization using quantum annealing on sparse Ising models. Front Phys 2:56

Bian Z, Chudak F, Macready WG, Rose G (2010) The Ising model: teaching an old problem new tricks. D-Wave Syst, 2

Brush Stephen G (1967) History of the Lenz-Ising model. Rev Mod Phys 39(4):883–893

Burer S, Monteiro R, Zhang Y (2002) Rank-two relaxation heuristics for max-cut and other binary quadratic programs. SIAM J Optim 12(2):503–521

Camsari KY, Faria R, Sutton BM, Datta S (2017) Stochastic p-bits for invertible logic. Phys Rev X 7(3):031014

Demir A, Mehrotra A, Roychowdhury J (2000) Phase noise in oscillators: a unifying theory and numerical methods for characterization. IEEE Trans Circ Syst I Fund Theory Appl 47(5):655–674

Denchev VS, Boixo S, Isakov SV, Ding N, Babbush R, Smelyanskiy V, Martinis J, Neven H (2016) What is the computational value of finite-range tunneling? Phys Rev X 6(3):031015

Ercsey-Ravasz M, Toroczkai Z (2011) Optimization hardness as transient chaos in an analog approach to constraint satisfaction. Nat Phys 7(12):966

Festa P, Pardalos PM, Resende MGC, Ribeiro CC (2002) Randomized heuristics for the MAX-CUT problem. Optim Methods softw 17(6):1033–1058

Gyoten H, Hiromoto M, Sato T (2018) Area efficient annealing processor for Ising model without random number generator. IEICE Trans Inform Syst E101.D(2):314–323

Gyoten H, Hiromoto M, Sato T (2018) Enhancing the solution quality of hardware Ising-model solver via parallel tempering. In: Proceedings of ICCAD'18, pp 70:1–70:8, New York, NY, USA, ACM

Hamerly R et al (2019) Experimental investigation of performance differences between coherent Ising machines and a quantum annealer. Sci Adv 5(5):eaau0823

Harris R, Johansson J, Berkley AJ, Johnson MW, Lanting T, Han S, Bunyk P, Ladizinsky E, Oh T, Perminov I et al (2010) Experimental demonstration of a robust and scalable flux qubit. Phys Rev B 81(13):134510

Helmberg C, Rendl F (2000) A spectral bundle method for semidefinite programming. SIAM J Optim 10(3):673–696

Hen I, Job J, Albash T, Rønnow TF, Troyer M, Lidar DA (2015) Probing for quantum speedup in spin-glass problems with planted solutions. Phys Rev A 92(4):042325

Hopfield JJ, Tank DW (1985) "Neural" computation of decisions in optimization problems. Biol Cybern 52(3):141–152

Inagaki T, Haribara Y, Igarashi K, Sonobe T, Tamate S, Honjo T, Marandi A, McMahon PL, Umeki T, Enbutsu K et al (2016) A coherent Ising machine for 2000-node optimization problems. Science 354(6312):603–606

Ising E (1925) Beitrag zur Theorie des Ferromagnetismus. Zeitschrift für Physik 31(1):253–258

Jensen Tommy R, Toft Bjarne (1995) Graph coloring problems. Wiley, New Jersey

Johnson MW, Amin MHS, Gildert S, Lanting T, Hamze F, Dickson N, Harris R, Berkley AJ, Johansson J, Bunyk P et al (2011) Quantum annealing with manufactured spins. Nature 473(7346):194

Karp RM (1972) Reducibility among combinatorial problems. Complexity of computer computations. Springer, Berlin, pp 85–103

King AD, Lanting T, Harris R (2015) Performance of a quantum annealer on range-limited constraint satisfaction problems. arXiv:1502.02098

Kuramoto Y (2003) Chemical oscillations, waves and turbulence. Dover, New York

Kuramoto Y (1975) Self-entrainment of a population of coupled non-linear oscillators. In: International symposium on mathematical problems in theoretical physics, pp 420–422. Springer

Lucas Andrew (2014) Ising formulations of many NP problems. Front Phys 2:5

Lucas A (2013) Ising formulations of many NP problems. arXiv:1302.5843

Lyapunov AM (1992) The general problem of the stability of motion. Int J Control 55(3):531–534

Mahboob I, Okamoto H, Yamaguchi H (2016) An electromechanical Ising Hamiltonian. Sci Adv 2(6):e1600236

Marandi A, Wang Z, Takata K, Byer RL, Yamamoto Y (2014) Network of time-multiplexed optical parametric oscillators as a coherent Ising machine. Nat Photon 8(12):937–942

Martí R, Duarte A, Laguna M (2009) Advanced scatter search for the max-cut problem. INFORMS J Comput 21(1):26–38

McMahon PL, Marandi A, Haribara Y, Hamerly R, Langrock C, Tamate S, Inagaki T, Takesue H, Utsunomiya S, Aihara K et al (2016) A fully-programmable 100-spin coherent Ising machine with all-to-all connections. Science 354:5178

Myklebust T (2015) Solving maximum cut problems by simulated annealing. arXiv:1505.03068

Neogy A, Roychowdhury J (2012) Analysis and design of sub-harmonically injection locked oscillators. In: Proceedings of IEEE DATE

Rinaldi G. Rudy graph generator code. https://www-user.tu-chemnitz.de/~helmberg/rudy.tar.gz. Website: https://www-user.tu-chemnitz.de/~helmberg/sdp_software.html

Robertson N, Sanders DP, Seymour P, Thomas R (1996) Efficiently four-coloring planar graphs. In: STOC '96: Proceedings of the 28th annual ACM symposium on theory of computing, pp 571–575. ACM

Rønnow TF, Wang Z, Job J, Boixo S, Isakov SV, Wecker D, Martinis JM, Lidar DA, Troyer M (2014) Defining and detecting quantum speedup. Science 345(6195):420–424

Sheldon F, Traversa FL, Di Ventra M (2019) Taming a nonconvex landscape with dynamical long-range order: memcomputing Ising benchmarks. Phys Rev E 100(5):053311

Shinomoto S, Kuramoto Y (1986) Phase transitions in active rotator systems. Prog Theor Phys 75(5):1105–1110

The G-set benchmarks for MAX-CUT. http://grafo.etsii.urjc.es/optsicom/maxcut

Wang T, Roychowdhury J (2017) Oscillator-based Ising machine. In arXiv:1709.08102

Wang T, Roychowdhury J (2014) PHLOGON: PHase-based logic using oscillatory nanosystems. In: Proceedings of Unconventional Computation and Natural Computation, LNCS sublibrary: Theoretical Computer Science and General Issues. Springer

Wang T (2017) Sub-harmonic injection locking in metronomes. arXiv:1709.03886

Wang T (2017) Achieving phase-based logic bit storage in mechanical metronomes. arXiv:1710.01056

Wang T, Roychowdhury J (2019) OIM: Oscillator-based Ising Machines for solving Combinatorial Optimisation problems. arXiv:1903.07163

Wang T, Roychowdhury J (2015) Design tools for oscillator-based computing systems. In: Proceedings of IEEE DAC, pp 188:1–188:6

Wang T, Wu L, Roychowdhury J (2019) New Computational Results and Hardware Prototypes for Oscillator-based Ising Machines. In: Proceedings IEEE DAC, pp 239:1–239:2

Wang T, Roychowdhury J (2019) OIM: Oscillator-based Ising Machines for Solving Combinatorial Optimisation Problems. In: Proceedings of International Conference on Unconventional Computation and Natural Computation

Winfree A (1967) Biological rhythms and the behavior of populations of coupled oscillators. Theor Biol 16:15–42

Yamamoto K, Huang W, Takamaeda-Yamazaki S, Ikebe M, Asai T, Motomura M (2017) A time-division multiplexing Ising machine on FPGAs. In: Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies, p 3. ACM

Yamaoka M, Yoshimura C, Hayashi M, Okuyama T, Aoki H, Mizuno H (2016) A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing. IEEE J Solid State Circ 51(1):303–309

Yin X, Sedighi B, Varga M, Ercsey-Ravasz M, Toroczkai Z, Hu XS (2018) Efficient analog circuits for Boolean satisfiability. IEEE Trans Very Large Scale Integr (VLSI) Syst 26(1):155–167