Title

Taming a nonconvex landscape with dynamical long-range order: Memcomputing Ising benchmarks

Journal

Physical Review E, 100(5)

ISSN

2470-0045

Authors

Sheldon, Forrest
Traversa, Fabio L
Di Ventra, Massimiliano

Publication Date

2019-11-01

DOI

10.1103/physreve.100.053311

Peer reviewed

Taming a non-convex landscape with dynamical long-range order: memcomputing the Ising spin-glass

Forrest Sheldon,[1,*] Fabio L. Traversa,[2,†] and Massimiliano Di Ventra[1,‡]

[1] *Department of Physics, University of California San Diego, La Jolla, CA 92093, USA*
[2] *MemComputing, Inc., San Diego, CA 92037, USA*

Recent work on quantum annealing has emphasized the role of collective behavior in solving optimization problems. By enabling transitions of large clusters of variables, such solvers are able to navigate their state space and locate solutions efficiently despite having only local connections between elements. However, collective behavior is not exclusive to quantum annealers, and classical solvers that display collective dynamics should also possess an advantage in navigating a non-convex landscape. Here, we propose a simple model that demonstrates this effect, based on the recently suggested digital memcomputing machines (DMMs), which utilize a collection of dynamical components with memory connected to represent the structure of the underlying optimization problem. This model, when applied to finding the ground state of the Ising spin glass, undergoes a transient phase of avalanches which can span the entire lattice. We then show that a full implementation of a DMM exhibits superior scaling compared to other methods when tested on the same problem class. These results establish the advantages of computational approaches based on collective dynamics.

Optimization problems draw their difficulty from the non-convexity of their associated landscapes [1]. These landscapes are often highly corrugated, dotted with hills, valleys and saddles of varying heights which obscure the search for a lowest (or highest) point. The complexity of this space, combined with the 'curse of dimensionality' yields an exponentially large number of potential solutions which are very difficult to prune down by any systematic method. The innate difficulty and variety displayed by optimization problems, as well as their widespread applications have made their study a continuously active field of research across science and mathematics [2, 3].

The exponential growth of the state space with problem size often renders any exact algorithm for locating the optimum impractical as they require an exponential amount of time to sift through the states. As a result, practitioners must rely on *incomplete* or *approximate* methods which will often generate better solutions in a limited time but are not guaranteed to converge to the exact solution [4, 5].

Early work on approximate methods relied on analogies with the dynamics of physical systems [6] which will minimize their energy as they cool, i.e., during annealing. For example, to find the ground state of the Ising spin glass [7],

$$E = -\sum_{\langle ij \rangle} J_{ij} s_i s_j, \quad s_i \in \{-1, 1\}, \tag{1}$$

simulated annealing gradually improves an initial state $\{s_i\}_{i=1}^N$ by stochastically exploring the state space and steadily lowering an effective temperature [8]. The early success of this approach on combinatorial optimization problems has led to the proliferation of solvers based on a similar stochastic local search and their many variants [9, 10]. Cross pollination with physics has continued, spawning methods such as parallel tempering [11], and quantum simulated annealing [12] as well as the analytical characterizations of combinatorial problems [13] and random energy surfaces [14].

Annealing has again jumped to the forefront of modern research in the form of quantum annealing and the machines manufactured by DWave [15–17]. These machines contain 2-state quantum mechanical elements coupled together in a graph realizing a particular energy function. During their relaxation, the quantum dynamics of the system allows for collective tunneling of elements through high, thin barriers in the energy function, which may provide some advantage in the search for the optimum.

Similar ideas in the context of cellular automata, neural networks and neuroscience have already received interest [18, 19]. These examples substantiate the idea that *collective behavior* would offer an advantage in the convergence of a solver by allowing for a more efficient exploration of the state space. We then expect that classical solvers which incorporate this feature in their dynamics will have an advantage in both the quality of solutions they produce, and their rate of convergence.

The purpose of this work is to explore the presence of collective dynamics in the context of specific deterministic dynamical systems: Digital memcomputing machines (DMMs) [20–22]. We show that this collective behavior, in the form of *dynamical long-range order* (DLRO), allows the efficient navigation of a non-convex landscape as the one provided by the prototypical Ising spin glass (9).

In DMMs, a combinatorial optimization problem is first transformed into a physical system described by differential equations whose equilibrium points corre-

* fsheldon@physics.ucsd.edu
† ftraversa@memcpu.com
‡ diventra@physics.ucsd.edu

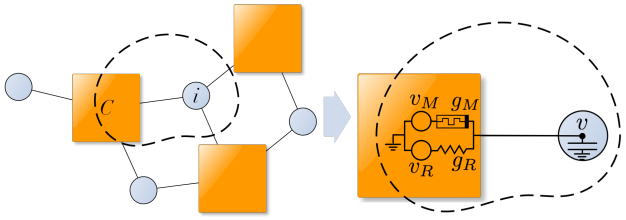

FIG. 1. **Constraint satisfaction problem as electrical circuit.** An arbitrary constraint (C) satisfaction problem expressed as a factor graph can be translated into an electrical circuit with memory by considering the effect of each constraint on the site i. v_R is a voltage generator and g_R is the conductance of a standard resistor. v_M is a voltage generator and g_M is the conductance of a resistor with memory.

spond to solutions of the original problem. Theoretical work [23] and simulations of DMMs [21, 22, 24, 25] have indicated the presence of long-range order in their dynamics. However, as their native problem form involves several distinct dynamical elements, the complexity of the resulting solver obscures the physical principles underlying its design and function. Here, by drawing on the structure of the equations governing a DMM, we propose a simplified model that captures their essential features and can be applied in a setting more familiar to physicists: finding the ground state of an Ising spin glass [7].

For our purposes, this problem provides the advantage that it can be expressed in terms of very simple homogeneous constraints leading to a concise set of equations. In addition, its real-space lattice representation allows for a clearer demonstration of DLRO since the real-space distance of the lattice corresponds to the distance in the constraint graph. We draw on a class known as 'frustrated-loop instances' used by DWave and others to benchmark their quantum annealers [15, 26]. These instances are constructed by embedding loops with a single frustrated bond on an underlying graph which we take to be a hypercubic lattice in 2 or 3 dimensions with periodic boundary conditions. These instances are convenient for benchmarking because, by construction, they have a known ground-state energy, and their difficulty is somewhat tunable by choosing the density of embedded loops [15]. The construction and tuning of these instances is detailed in the Supplemental Information. After showing DLRO in the simplified model of DMMs, we then compare the results obtained by a full-fledged implementation of DMMs with those from a variety of approaches, such as parallel tempering [11], simulated annealing [6], as well as a commercial solver [27], for problem instances of increasing size. We find that DMMs exhibit superior scaling compared to these other methods.

A DMM is constructed in correspondence to the logical circuit it will solve. For example, the subset-sum problem studied in [21] utilizes a circuit with the same structure as one used to add a subset from a group of numbers. Each traditional logic gate is replaced by a *self-organizing logic gate* consisting of a set of interconnected input and output terminals, each of which is dressed with a number of memristors (resistors with memory), resistors, capacitors, and voltage/current generators forming a *dynamic correction module* (DCM) [21]. When voltages are applied to the boundaries of the circuit, the dynamics of these elements are configured to satisfy the constraints enforced by each gate, and lead the circuit to a state where no logical contradictions are present.

We may consider the contribution of constraint C to the dynamics of site i (see Fig. 1) [21]. The dynamics of the circuit are constructed such that the voltage generators impose the logical constraint on the voltage v_i at site i. The memristor conductance g_M, sensing a current flowing across it due to an unsatisfied constraint, will alter its value to accelerate the convergence of v_i to the logically-consistent solution. Generally, this is accomplished by increasing the memristor conductance, thus allowing more current to flow into or out of the site. As memristors are polar objects, complex constraints may require several memristors and generators to accomplish this, accounting for the number of memristors in DCMs [21].

A few simplifying assumptions give the general form for the contribution of constraint C to site i as [21],

$$\dot{v}_i = \Delta g_M x \Delta V_M + g_R \Delta V_R, \tag{2}$$

$$\dot{x} = h(\Delta V_M, x), \quad x \in [0,1], \tag{3}$$

for the voltage v_i representing the variable i and the memory state variable of the memristor x. We can regard the first and second terms on the rhs of Eq. (2) as representing the *total* memrisitive and resistive contributions from the DCM, respectively. These are weighted by the conductances Δg_M and g_R, respectively, into which we have absorbed a capacitive timescale. We regard the memory state variable x and function h in Eq. (3) as an effective representation of the state and evolution of all memristors in the DCM, giving us considerable freedom in choosing the form of h. In order to fulfill the requirements of a DMM, the memristor equations chosen must take on *bounded* values and the equations of motion of the whole system must be *point-dissipative* [28], which establishes that trajectories will converge to an invariant set that is uniformly asymptotically stable.

These equations share a close resemblance to those of Lagrange programming neural networks (LPNNs) proposed in [29, 30] and the dynamical systems proposed in [31]. In these works a Lagrangian, $\mathcal{L}$, for a constraint satisfaction problem on variables $\{s_i\}$ is formed from a set of constraint functions $C_m(\{s_i\})$ which vary from 0 when the constraint is satisfied to 1 when unsatisfied and a set of weights for each constraint x_m, $\mathcal{L} = \sum_m x_m C_m(\{s_i\})$. In the case of LPNNs, the equations of motion of the system are then derived as

$$\dot{s}_i = -\nabla_{s_i}\mathcal{L} = -\sum_m x_m \nabla_{s_i} C_m, \tag{4}$$

$$\dot{x}_m = \nabla_{x_m}\mathcal{L} = C_m, \tag{5}$$

which in our formulation (Eqs. (2) and (3)) would correspond to an *unbounded*, voltage-controlled set of memristors with equal weight. In [31] the equations for the multipliers are altered to $\dot{x}_m = x_m C_m$, which has the effect of making the system hyperbolic, and is analogous to choosing *unbounded* current-controlled memristors in Eq. (3). The dynamics of both systems are such that the variables s_i of the optimization problem act to minimize the energy, while the weights x_m act to increase it, forming a sort of competitive dynamics which seek out saddle points in the Lagrangian. The weights may be re-expressed as a memory term in the s_i equations and so may be interpreted as "memory variables."

The continuous constraint weighting that these Lagrangian methods perform bears a close resemblance to DMMs, but in our investigations we do not observe DLRO in the simulations of these Lagrangian systems nor do they reach the solution of the problem we consider here. Instead, in order to fulfill the properties of a DMM and display the DLRO observed in these, additional terms in the equations of motion are necessary, in particular terms that guarantee the orbits are bounded during dynamics, and that the system manifests a form of "rigidity" in which large groups of variables can transition together (for further discussion of rigidity in a continuous dynamical model of the spin glass, see the Supplemental Information).

When applied to finding the ground state of the Ising spin glass, Eq. (9), a simple representation of DMMs (Eqs. (2) and (3)) that satisfies these requirements is of the form,

$$\dot{v}_i = \frac{1}{2}\sum_{\langle ij\rangle} |J_{ij}|x_{ij}\big(v_i + \mathrm{sgn}(J_{ij})v_j\big) - |J_{ij}|\big(v_i - \mathrm{sgn}\big(J_{ij}\big)v_j\big), \quad v_i \in [-1,1], \tag{6}$$

$$\dot{x}_{ij} = \beta x_{ij}(1-x_{ij})\Big(|J_{ij}|\big(1-\mathrm{sgn}(J_{ij})v_i v_j\big) - \gamma\Big), \tag{7}$$

in which the memristive and resistive contributions are clearly visible. The voltages v_i are limited to the interval $[-1,1]$. From a state of the dynamical system, the spins of the original Ising spin glass model (9) are assigned as $s_i = \mathrm{sgn}(v_i)$ such that the spins of the Ising model undergo the orthant dynamics of the underlying continuous voltages.

The memory state follows the simplest equation for a *bounded*, volatile memristor subject to an effective voltage $|J_{ij}|(1-\mathrm{sgn}(J_{ij})v_i v_j)$. This voltage is the energy with which the constraint is violated, and the constant γ sets a threshold below which x_{ij} will begin to decay. The constant β indicates that the memristive timescale is generally different from the voltage timescale (set by the RC constant at the node) which will play an important role in our analysis of the system. It is useful to re-express (6) in terms of the regimes of the memory variable x_{ij},

$$\dot{v}_i = \sum_{\langle ij\rangle} J_{ij}x_{ij}v_j - (1-x_{ij})\frac{|J_{ij}|}{2}(v_i - \mathrm{sgn}(J_{ij})v_j), \tag{8}$$

which shows that the x_{ij} interpolate between two different interactions between the voltages. When $x_{ij} \approx 1$ the voltages follow the fields imposed by the neighboring voltages as in an LPNN with $\mathcal{L} = \sum_{\langle ij\rangle} x_{ij}|J_{ij}|(1 - \mathrm{sgn}(J_{ij})v_i v_j)$, causing them to take the integral values $v_i = \pm 1$. Once the constraint is satisfied, $x_{ij} \to 0$, and the voltages follow the values of their neighbors in a collective manner (see also the Supplemental Information). As a consequence, over the course of the dynamics, voltages form clusters with satisfied constraints that are capable of transitioning together under the influence of neighboring unsatisfied constraints. This has a dramatic effect on the dynamics and inclusion of these "rigidity terms" to the gradient-like first terms in Eq. (6) are essential for achieving DLRO in the dynamics and converging to the ground state.

We simulate the system described by Eqs. (2) and (3) from random initial voltages and $x_{ij}(0) = 0.99$, integrating the equations of motion until the energy (9) (calculated from the signs of the voltages) has reached the planted ground state or some maximum time has elapsed. This is typically chosen quite long, such that the system solves an instance with a probability $p > 0.95$ for a given initial condition. For a more detailed discussion of the numerical implementation, see the Supplemental Information. A typical run, showing the voltages, memristances and energy of the system is shown in Fig. 2 on a 2-dimensional instance, $L = 15$, where we also show that in the absence of constraint weighting via the memory variables ($\dot{x}_{ij} = 0$) the system is unable to reach the ground state (the red curve in Fig. 2(c)). In this case the system undergoes gradient dynamics and converges to a local minimum of $\mathcal{H} = \sum_{\langle ij\rangle} -J_{ij}v_i v_j$, $v_i \in [-1,1]$. The action of the memory variables may be interpreted as slowly modifying this landscape to destabilize these local minima and push the system into an avalanche. That these avalanches display DRLO, is a feature of the added "rigidity terms" in Eq. (6).

The discussion of DLRO in continuous dynamical systems is complicated by the continuity of the dynamics, making it difficult to clearly infer causal relationships between changes in variables. However, we can take advantage of the timescales above to separate the dynamics into causally related events. As shown in Fig. 2(a) when we slow the memristor timescale β relative to that of the voltages (e.g, by choosing $\beta = 1/400$), after the initial transient the dynamics progress through a series of rapid transitions interpretable as *avalanches* (or instantons [23]). These are due to the gradient dynamics of the voltages rapidly seeking out saddle points in the energy landscape. Since the memristive dynamics provide the unstable directions of the saddle points [22, 23], the system will rapidly shift to a new saddle point. As

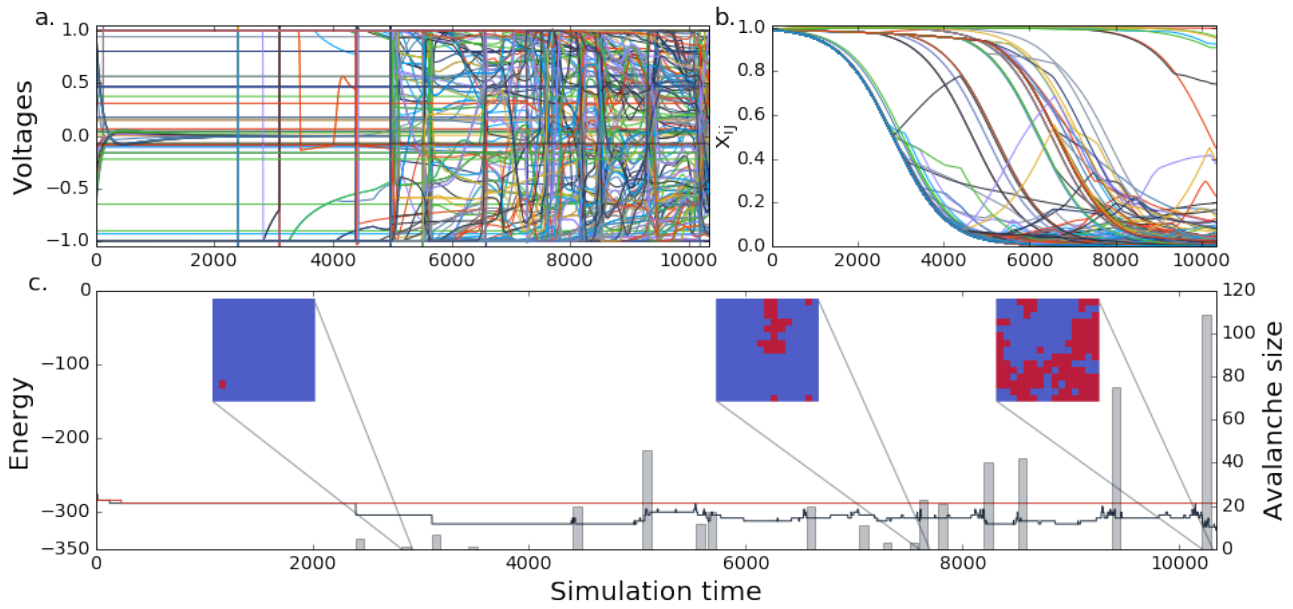

FIG. 2. **Model trajectories.** When the model of Eqs. (6) and (7) is simulated for a 2-dimensional instance ($L = 15$) under a separation of timescales ($\beta = \frac{1}{400}$), the voltage trajectories (a) evolve under a series of sharply defined avalanches due to the slow motion of the memristors (b) modifying the clause weights. In (c) we have plotted the energy (left axis) without the influence of memristors (red, $\beta = 0$) and with them (black, $\beta = \frac{1}{400}$, $\gamma = 0.65$) showing that the motion of the memory variables allows the system to reach a far lower energy, and ultimately the ground state. The sizes of the avalanches (c, right axis) are plotted as gray bars, showing that their size grows over the course of the simulation until a large avalanche brings the system to its ground state. The avalanches are depicted in the inset in red with the rightmost inset corresponding to the largest avalanche in the run.

more constraints become satisfied and transition to a rigid interaction, larger clusters of voltages begin transitioning together (see Fig. 2(c)) until, in a manner analogous to a phase transition [22, 32], extensive clusters of voltages/spins, spanning the *entire* lattice, transition *collectively* and the system converges to the ground state.

We can further emphasize the long-range nature of these clusters by computing correlation functions over the course of an avalanche. In the limit that the timescales become separated (i.e., the slow driving limit) the points at which each avalanche occurs tend towards well defined times as seen in Fig. 2(a). For small β these events may be detected as sharp spikes in the voltage derivatives. (See Supplemental Information for a detailed discussion of the method used to extract the structure of the avalanches.) We are interested in the voltages/spins which change sign in the avalanche and thus will affect the energy of the system. We thus define the avalanche configuration as $\Delta_i = 1$ for all spins which change sign during an avalanche, and $\Delta_i = 0$ otherwise. A few typical examples of these avalanches and their sizes occurring during dynamics are plotted in Fig. 2(c).

Using the avalanche configurations we are able to compute correlation functions for these events and investigate their decay across the lattice. For each run (defined as generating a unique instance and initial conditions) the system is simulated until it reaches the ground state or a maximum time is reached. If the instance is solved within this interval, the largest avalanche is selected and

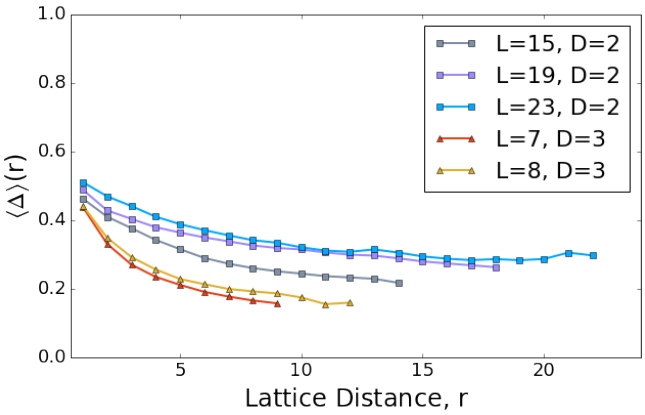

FIG. 3. **Long-range order.** Spatial correlations, $\langle\Delta\rangle(r)$, among voltages/spins calculated from the orthant dynamics in the slow driving limit of model (6) and (7) for the largest avalanches in 2D and 3D and for different lattice sizes. The correlations take a finite value all the way to the lattice edge, indicating that the largest avalanches are extensive. As the system size increases the values appear to saturate to a dimension-dependent value for this instance class.

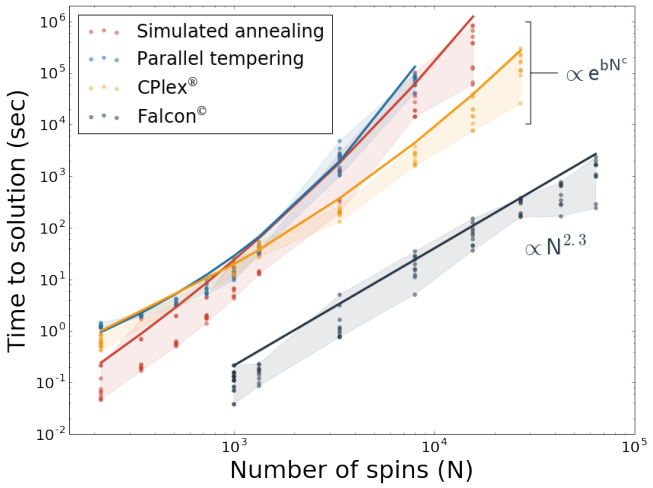

FIG. 4. **Time to solution.** Time necessary for different solvers to reach the ground state of the 3D frustrated-loop spin glass as a function of the total number of spins N. The sequential memcomputing solver implemented in MATLAB is dubbed Falcon. Ten instances for each problem size are displayed. Comparisons with simulated annealing (SA), parallel tempering (PT), and IBM CPlex are also shown. All calculations were performed on a single core. The solid lines are the best fits of the worst-case time to solution for all four solvers. The exponential fit has the following parameters: for IBM CPlex, $b = 4.5$ and $c = 0.16$, for SA, $b = 11.$ and $c = 0.13$, and for PT $b = 0.9$ and $c = 0.33$.

its configuration and first flipping spin are stored. By averaging across a sample of configurations, suitably shifted so that the initial flipping spins coincide, the probability that a voltage a distance r from the initial spin changes sign, $\langle\Delta\rangle(r)$, may then be calculated. In order to achieve large distances with reasonable simulation times, we calculated these correlations both in 2- ($L = 15, 19, 23$) as well as 3-dimensional ($L = 7, 8$) systems.

As shown in Fig. 3, the largest avalanches possess correlations that take finite values all the way to the furthest corner of the lattice, confirming the presence of DLRO. Dimensionally, this requires that the size of the largest avalanche scales as $\sim L^D$ for a system of dimension D, and is thus extensive. We also note that, as the system size increases the correlations appear to saturate to a dimension (and instance class) dependent value.

Having used this simplified model to demonstrate the main physical ingredients promoting DLRO, we now show that this feature also leads to superior scaling compared to other methods that rely only on local information. Unlike the model (6) and (7), DMMs have the additional advantage of having been engineered to have no chaotic behavior or periodic orbits [33]. Therefore, for this scalability test, we rely on the full implementation of the dynamical equations of DMMs as in Ref. [21], appropriately modified to handle the Ising spin glass expressed as a maximum satisfiability problem in conjunctive normal form [34] (see the Supplemental Information for a discussion of this transformation). We then utilize a commercial sequential MATLAB solver (dubbed Falcon) that implements such equations. In addition, we have implemented two standard annealing algorithms in Python (simulated annealing (SA) and parallel tempering (PT)), as well as used a well-known commercial mixed-integer programming solver, IBM CPlex [27]. Since Falcon was implemented in interpreted MATLAB and the focus was on scaling rather than runtime, we used only the simplest implementation of each solver but performed substantial tuning. Details of the implementation and tuning on the instance class for SA and PT, as well as the configuration for IBM CPlex can be found in the Supplemental Information.

All solvers were run on 10 frustrated-loop instances in 3 dimensions, ranging in size from $L = 6$ (total number of spins $N = 216$) to $L = 40$ ($N = 64,000$). As is clearly visible from Fig. 4, Falcon converged to the exact ground state in times orders of magnitude faster than the other methods tested, and for larger sizes than were attainable for other solvers even given one week of computing time. Most importantly, Falcon displays superior scaling, with the time to solution (TTS) appearing to scale approximately as $TTS \sim N^{2.3}$, while all other instances appear to scale exponentially, $TTS \sim \exp(bN^c)$, with b and c solver-specific constants reported in the supplemental information and shown in Fig. 4. Of the solvers tested, SA appears closest to subexponential. However, simulated annealing has been shown to scale exponentially on a closely related instance class [15] and, as all solvers, except Falcon, were unable to converge for all instances within the allotted time (1 week), the reported exponential fits may underestimate their actual scaling. Details of the fitting procedure may be found in the Supplemental Information.

Conclusions – In this paper, using the Ising spin glass as a well-known benchmark, we have shown that a solver exploiting dynamical long-range order can navigate a non-convex landscape more efficiently than traditional methods. We have first provided a simple model of DMMs to show how to transform the original problem into a dynamical system in which DLRO emerges naturally. We have then shown results on the 3D Ising spin-glass as obtained by a full implementation of DMMs. The approach based on DMMs exhibits superior scaling in reaching the solution than the other methods tested. The results presented here further reinforce the advantages of employing collective dynamics to compute hard problems efficiently.

Acknowledgments – F.S. and M.D. acknowledge partial support from the Center for Memory and Recording Research at UCSD. The Falcon solver used in the reported simulations has been provided by MemComputing, Inc. http://memcpu.com/. The authors would be delighted to provide, upon request, all instances of the spin-glass problems used in this work.

[1] Moore, C. & Mertens, S. *The Nature of Computation* (Oxford University Press, Inc., New York, NY, USA, 2011).

[2] Christos H. Papadimitriou, K. S. *Combinatorial Optimization* (Dover Publications Inc., 1998).

[3] Edwin K. P. Chong, S. H. Z. *An Introduction to Optimization* (John Wiley & Sons Inc, 2013).

[4] Kautz, H. A., Sabharwal, A. & Selman, B. Incomplete algorithms. In Biere, A., Heule, M. & van Maaren, H. (eds.) *Handbook of satisfiability*, vol. 185, 185–203 (IOS press, 2009).

[5] Gomes, C. P., Kautz, H., Sabharwal, A. & Selman, B. Satisfiability solvers. In Van Harmelen, F., Lifschitz, V. & Porter, B. (eds.) *Handbook of knowledge representation*, vol. 1, chap. 2, 89–134 (Elsevier, 2008).

[6] Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).

[7] Fischer, K. H. & Hertz, J. A. *Spin glasses*, vol. 1 (Cambridge University Press, 1993).

[8] Cocco, S., Monasson, R., Montanari, A. & Semerjian, G. Analyzing search algorithms with physical methods. In Percus, A., Istrate, G. & Moore, C. (eds.) *Computational complexity and statistical physics* (OUP USA, 2006).

[9] Selman, B., Levesque, H. J., Mitchell, D. G. *et al.* A new method for solving hard satisfiability problems. In *AAAI*, vol. 92, 440–446 (Citeseer, 1992).

[10] Schoning, T. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, 410–414 (1999).

[11] Wang, W., Machta, J. & Katzgraber, H. G. Comparing monte carlo methods for finding ground states of ising spin glasses: population annealing, simulated annealing, and parallel tempering. *Phys. Rev. E* **92**, 013303 (2015).

[12] Santoro, G. E., Martoňák, R., Tosatti, E. & Car, R. Theory of quantum annealing of an ising spin glass. *Science* **295**, 2427–2430 (2002).

[13] Mézard, M., Parisi, G. & Zecchina, R. Analytic and algorithmic solution of random satisfiability problems. *Science* **297**, 812–815 (2002).

[14] Bray, A. J. & Dean, D. S. Statistics of critical points of gaussian fields on large-dimensional spaces. *Phys. Rev. Lett.* **98**, 150201 (2007).

[15] Hen, I. *et al.* Probing for quantum speedup in spin-glass problems with planted solutions. *Phys. Rev. A* **92**, 042325 (2015).

[16] Denchev, V. S. *et al.* What is the computational value of finite-range tunneling? *Phys. Rev. X* **6**, 031015 (2016).

[17] Katzgraber, H. G., Hamze, F., Zhu, Z., Ochoa, A. J. & Munoz-Bauza, H. Seeking quantum speedup through spin glasses: The good, the bad, and the ugly. *Phys. Rev. X* **5**, 031026 (2015).

[18] Langton, C. G. Computation at the edge of chaos: phase transitions and emergent computation. *Physica D* **42**, 12–37 (1990).

[19] Chialvo, D. R. Emergent complex neural dynamics. *Nat. Phys.* **6**, 744 (2010).

[20] Traversa, F. L. & Di Ventra, M. Universal memcomputing machines. *IEEE Trans. Neural Networks Learn. Syst.* **26**, 2702 (2015).

[21] Traversa, F. L. & Di Ventra, M. Polynomial-time solution of prime factorization and np-complete problems with digital memcomputing machines. *Chaos* **27**, 023107 (2017).

[22] Di Ventra, M. & Traversa, F. L. Memcomputing: Leveraging memory and physics to compute efficiently. *J. Appl. Phys.* **123**, 180901 (2018).

[23] Di Ventra, M., Traversa, F. L. & Ovchinnikov, I. V. Topological field theory and computing with instantons. *Ann. Phys. (Berlin)* **529**, 1700123 (2017).

[24] Traversa, F. L., Cicotti, P., Sheldon, F. & Di Ventra, M. Evidence of exponential speed-up in the solution of hard optimization problems. *Complexity* **2018**, 7982851 (2018).

[25] Manukian, H., Traversa, F. L. & Di Ventra, M. Accelerating deep learning with memcomputing. *arXiv:1801.00512* (2018).

[26] King, A. D., Lanting, T. & Harris, R. Performance of a quantum annealer on range-limited constraint satisfaction problems. *arXiv preprint arXiv:1502.02098* (2015).

[27] Ibm ilog cplex optimizer. URL www-01.ibm.com/software/integration/optimization/cplex-optimizer.

[28] Hale, J. *Asymptotic Behavior of Dissipative Systems*, vol. 25 of *Mathematical Surveys and Monographs* (American Mathematical Society, Providence, Rhode Island, 2010), 2nd edn.

[29] Zhang, S. & Constantinides, A. Lagrange programming neural networks. *IEEE Trans. Circuits Syst. II* **39**, 441–452 (1992).

[30] Nagamatu, M. & Yanaru, T. On the stability of lagrange programming neural networks for satisfiability problems of prepositional calculus. *Neurocomputing* **13**, 119–133 (1996).

[31] Ercsey-Ravasz, M. & Toroczkai, Z. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nat. Phys.* **7**, 966 (2011).

[32] Sheldon, F. C. & Di Ventra, M. Conducting-insulating transition in adiabatic memristive networks. *Phys. Rev. E* **95**, 012305 (2017).

[33] Di Ventra, M. & Traversa, F. L. Absence of periodic orbits in digital memcomputing machines with solutions. *Chaos* **27**, 101101 (2017).

[34] Garey, M. R. & Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., New York, NY, USA, 1990).

[35] Mandra, S., Katzgraber, H. G. & Thomas, C. The pitfalls of planar spin-glass benchmarks: raising the bar for quantum annealers (again). *Quantum Science and Technology* **2**, 038501 (2017).

[36] Barahona, F. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General* **15**, 3241 (1982).

[37] Peskin, M. E. & Schroeder, D. V. *An Introduction to Quantum Field Theory* (Westview Press, 1995).

[38] Morris, P. The breakout method for escaping from local minima. In *AAAI 93* (1993).

[39] Selman, B. & Kautz, H. Domain-independent extension to gsat: Solving large structured satisfiability problems. In *Proc. IJCAI-93*, 290–295 (1993).

[40] Jia, H., Moore, C. & Selman, B. From spin glasses to

hard satisfiable formulas. In *International Conference on Theory and Applications of Satisfiability Testing*, 199–210 (Springer, 2004).

[41] Jones, E., Oliphant, T., Peterson, P. *et al.* SciPy: Open source scientific tools for Python (2001–). URL http://www.scipy.org/. [Online; accessed ¡today¿].

[42] Sheldon, F., Cicotti, P., Traversa, F. L. & Di Ventra, M. Stress-testing memcomputing on hard combinatorial optimization problems. *arXiv:1807.00107* .

Supplemental Information: Taming a non-convex landscape with dynamical long-range order: Memcomputing the Ising spin glass

I. CORRESPONDENCE BETWEEN STATISTICAL PHYSICS AND COMBINATORIAL OPTIMIZATION

The problem of finding the ground state of a system in statistical physics is an optimization problem for which there is an extensive vocabulary in computer science [1]. It may be useful for the reader familiar with physics to have some notion of this correspondence and we include a short discussion here to that effect.

A Hamiltonian expressed as a sum over interactions between spins may be expressed as a constraint satisfaction problem where each term in the Hamiltonian is regarded as a constraint on the variables. For example, the Ising Model

$$E = -\sum_{\langle ij \rangle} J_{ij} s_i s_j, \quad s_i \in \{-1, 1\}, \tag{9}$$

is equivalent to a weighted constraint satisfaction problem, where each interaction is expressed as an exclusive-OR (XOR), or sum modulo-2 between the associated binary variables, $b_i = (s_i + 1)/2$, $j_{ij} = (1 - \mathrm{sgn}(J_{ij}))/2$

$$-J_{ij} s_i s_j \quad \leftrightarrow \quad 2|J_{ij}|\; b_i \oplus b_j = j_{ij} \tag{10}$$

where $2|J_{ij}|$ is the weight associated with the constraint. The correspondence between these two can also be easily seen from the fact that flipping the state of any variable changes the state of the interaction in both cases. Transformations between other constraint satisfaction problem types may be undertaken similarly. For example, when transforming to weighted conjunctive normal form (CNF), each interaction may be translated to two OR constraints depending on the sign of the interaction:

$$-J_{ij} s_i s_j \quad \leftrightarrow \quad \begin{cases} 2|J_{ij}|\; b_i \vee \bar{b}_j \\ 2|J_{ij}|\; \bar{b}_i \vee b_j, \end{cases} \quad \mathrm{sgn}(J_{ij}) = 1, \qquad \begin{cases} 2|J_{ij}|\; b_i \vee b_j \\ 2|J_{ij}|\; \bar{b}_i \vee \bar{b}_j, \end{cases} \quad \mathrm{sgn}(J_{ij}) = -1, \tag{11}$$

where each constraint carries a weight and negations are indicated with a bar, e.g., $\bar{b}$. In all cases, the factor of 2 may be dropped as a global scaling of the energy.

Constraint satisfaction *instances* (a particular example of the problem) may be described as being either satisfiable (SAT), if there is an assignment of the variables which satisfies every constraint, or unsatisfiable (UNSAT) if there is no satisfying assignment (commonly referred to as *frustrated* in physical treatments).

The corresponding *decision problem* of determining whether such an assignment exists, and, therefore, whether a particular instance is SAT or UNSAT is also referred to as SAT or satisfiablity with context generally

determining which meaning is intended. The *optimization problem* of determining an assignment which satisfies the maximum number of constraints (or maximum total weight) is referred to as MAXSAT. Determining the ground state of a system in statistical physics is thus equivalent to a MAXSAT instance.

Generally, the SAT problem on 2-variable OR and XOR constraints may be trivially solved. In the case of the Ising model, pick the value of any spin to be +1 and propagate this throughout the lattice where every spin value will be determined by its neighbor. If a contradiction is reached, the instance is unsatisfiable. If not, this will construct a satisfying assignment and the instance is equivalent to the ferromagnetic Ising model through a gauge transformation.

Despite this, the MAXSAT problem on two variable constraints may be quite difficult, depending on the structure of the instance. It is known that instances on a planar graph may be solved efficiently (in polynomial time) by a perfect matching algorithm [35]. If the graph is non-planar as in the chimera graphs used by DWave [16] or the 3-dimensional cubic lattice used for benchmarking here, there is no general efficient algorithm known, and the problem of finding an assignment is NP-hard [36]. This statement, however, only applies to the worst cases and for any individual instance, and especially for classes of randomly generated instances, one might hope that an efficient approach exists. Conversely, despite the fact that an efficient algorithm exists for planar instances, they may still present meaningful difficulty for a solver which only uses local information. Debate over these ideas have surrounded the benchmarking studies for DWave and discussions to this effect may be found in [15, 17, 26, 35].

II. 'RIGIDITY' IN A CONTINUOUS DYNAMICAL SYSTEM

The notion of 'rigidity' arises in several areas across physics and here we clarify what our intended meaning is in the context of continuous dynamical systems. The continuity of these systems can give rise to behaviors inaccessible to their discrete counterparts [22, 23]. In particular, the presence of a continuous symmetry in the equations and its effective breakdown can give rise to behavior analogous to zero-modes in statistical physics/field theory [37]. As a consequence, along some directions of the phase space the system can respond in a correlated, or 'rigid' manner in which large clusters of variables will transition together [23].

For example, in a lattice of continuous "spins" obeying,

$$\dot{v}_i = -|J_{ij}|\big(v_i - \mathrm{sgn}(J_{ij})\sigma(v_j)\big), \tag{12}$$

$$\sigma(x) = \begin{cases} 1, & x > 1 \\ x, & -1 \le x \le 1 \\ -1 & -1 < x. \end{cases} \tag{13}$$

the system will exponentially relax to a state in which every variable v_i takes the value $\mathrm{sgn}(J_{ij})v_j$ for all of its neighbors v_j. If the underlying lattice is ferromagnetic ($J_{ij} = 1$), then taking any spin to its limiting value $v_i = \pm 1$ will cause the entire lattice to transition with it in a manner analogous to long-range order. In contrast, for a discrete system in the ferromagnetic state $s_i = 1$, flipping a single spin will not cause the rest of the lattice to transition as it will not change the sign of any local fields. The ability of a local perturbation to flip large clusters of spins might benefit a solver attempting to satisfy a constraint while maintaining the satisfaction of its neighbors. However, the presence of unsatisfiable/frustrated constraints renders this impossible in the model above: in this case all spins will relax to $v_i = 0$ and pulling a single spin to ± 1 will not propagate through the lattice.

Any unsatisfiable spin-glass instance may be associated with one or several satisfiable instances formed by removing any unsatisfied bonds in the ground state (see the preceding Sec. I for a discussion of these terms in the context of computer science). In the main text we show that the inclusion of memory variables to the dynamical system (12) above restores some measure of the long-range order. These variables come in the form of constraint weights which act to isolate the associated satisfiable instance and bear similarities to strategies employed in discrete constraint satisfaction [38, 39].

III. GENERATING INSTANCES: FRUSTRATED LOOP INSTANCES AND INSTANCE TUNING

The problem of benchmarking MAXSAT solvers is generally hindered by the fact that the problems are NP-hard, and, for an arbitrary instance, determining or even confirming a solution will require exponential time [36]. For this reason, planted solution instances are commonly employed in which instances are generated such that they have a known solution [40].

Benchmarking studies on quantum annealing have introduced the class of Frustrated Loop Hamiltonians [15, 26] in which the total Hamiltonian is written as the sum of a set of loops containing a single frustrated bond (see schematic in Fig. 5),

$$H = \sum_i H_{FL,i}. \tag{14}$$

The loops are formed such that the planted solution minimizes all of the Hamiltonians $H_{FL,i}$ simultaneously, and so minimizes their sum.

In order to generate these instances, we first construct an underlying lattice which we take to be hypercubic in D-dimensions with periodic boundary conditions. Each loop is generated by beginning at a randomly selected site and performing a random walk until it crosses itself. The length, l, of the loop formed is generally required to be above some limit, otherwise it is rejected. For instance,

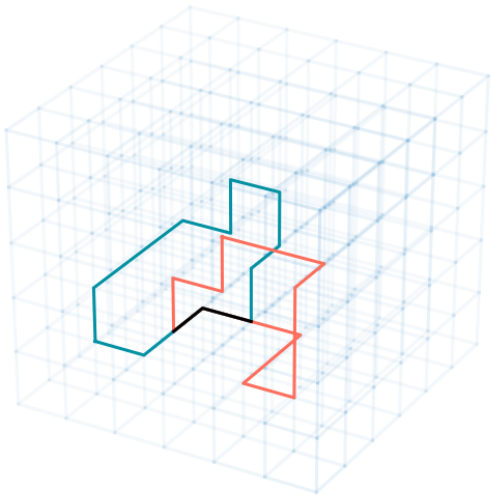

FIG. 5. **A schematic representation of instance creation.** Separate frustrated loops (blue and red curves) are generated by random walking around the lattice until the walk crosses itself. Each loop has its own Hamiltonian consisting of $J_{ij} = 1$ for all bonds except one with $J_{kl} = -1$ such that the ground state of the loop will have one unsatisfied bond. When the loops are combined, overlapping bonds (shown in black) have a coupling J_{ij} which is the sum of the contributions from each separate loop.

the instances used by DWave use a loop length limit of $l \geq 8$ [15, 16]. It is also noted that discarding the length limit seems to lead to very difficult instances, although an explanation for this feature is not understood. In our investigations of the instances, we found that discarding the loop length limit leads to instances of widely varying difficulty, and that both the uniformity of the difficulty, and the time to solution (measured with simulated annealing) decreased as the length limit increased. In order to avoid the complications of a widely varying difficulty, while generating the most difficult available instances, we chose a length limit of $l \geq 6$ for our generated instances.

In order to generate a loop, we consider planting the ferromagnetic solution $s_i = 1$. After generating an instance, any other solution may be hidden by means of a gauge transformation. All interactions in the loop are chosen to be ferromagnetic, $J_{ij} = 1$, except one which is selected at random to be anti-ferromagnetic $J_{ij} = -1$. The solution to the loop hamiltonian $H_{FL,i} = -\sum_{\langle ij \rangle \in l_i} J_{ij} s_i s_j$ is thus an assignment with one unsatisfied interaction.

The number of loops, M, generated must be proportional to the number of sites $N = L^D$ and may be characterized by a *density* α such that $M = \alpha N$. These instances are known to demonstrate a *hardness peak* in α such that the most difficult instances are generated when there are neither too few loops, in which case they do not overlap and each may be solved separately, nor too many, in which case the antiferromagnetic interactions tend to be canceled by the more numerous ferromagnetic interactions [15–17]. The value of α at the peak also tends to align with the amount of frustration in the instance, as measured by the number of unsatisfied interactions in the ground state.

In order to generate difficult instances, in $D = 2$ dimensions we used a simulated annealing solver to test instances across a range of α, finding that the most difficult instances lay at $\alpha \approx 0.2$, consistent with the results on the pseudoplanar chimera graphs in [15]. For $D = 3$ dimensions, the optimal value of α was estimated using the amount of frustration in the instances as suggested in [15] and found to lie at $\alpha \approx 0.3$.

IV. AVALANCHE DETECTION AND CORRELATION CALCULATIONS

As shown in the main text, when the timescale of the memristors is sufficiently separated from the voltage timescale, the system evolves in well defined events which may be interpreted as avalanches (or instantons [23]). A set of sample trajectories, calculated on different spin glass instances ($L = 15$, $D = 2$, $\alpha = 0.2$) in the slow driving limit is displayed in Fig. 6. Within these trajectories avalanches are clearly discernible, but the timescales governing them seem to grow during the simulation and additional features such as quasi-periodic behavior (as seen in panel 6.c at beginning time 8000) also arise. Detecting these events in a robust manner poses a novel problem as we must extract a discrete event from a continuous system. Here we detail the method we used to extract these events and compute correlations during the largest avalanche.

The rate of change of the entire system may be concisely viewed through the magnitude of the voltage derivative vector, $\dot{\vec{v}}(t)$, normalized to the number of spins, $|\dot{\vec{v}}(t)|/N$ shown in Fig. 7. Avalanches manifest as sharp spikes in the magnitude of the derivative. However, as the simulation advances and variables begin transitioning together, the slowest timescale in the system tends to increase, making a simple threshold ineffective at separating the later clusters.

Instead, we first find the convex lower envelop of the total derivative shown in Fig. 7. This gives an estimate of how the slowest timescale in the system changes. If the slope at the end of the envelope exceeds a bound, its slope is extrapolated from the previous point to avoid errors due to the termination of the integration mid-avalanche.

The time interval of an avalanche is defined as a continuous period in which the magnitude of the derivative $|\dot{\vec{v}}(t)|/N$ exceeds the lower envelop by a threshold. Choosing this threshold is performed through tuning to the specific set of instances and will depend on system size, dimension and memristor timescale β. The threshold value t used for each set of correlation calculations

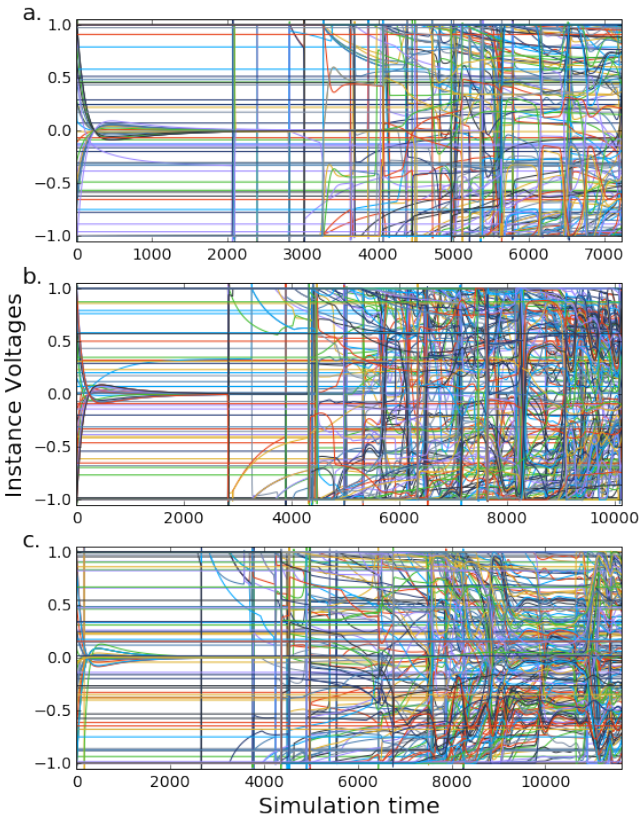

FIG. 6. **A sample of trajectories.** Here we show the simulation results for the same initial conditions on three different instances ($L = 15$, $D = 2$, $\alpha = 0.2$). In (a.) we see well separated events that maintain their separation until the instance is solved. In (b.) the longer run results in lower memristor values and a slower voltage timescale, causing the width of the avalanches to grow. In (c.) we see markers of quasi-periodic behavior extending from ≈ 8000 to 10000. A scheme to detect avalanches must be robust to these effects in order to be accurate.

are: ($L = 15$, $D = 2$, $t = 1 \times 10^{-4}$), ($L = 19$, $D = 2$, $t = 2.15 \times 10^{-5}$), ($L = 23$, $D = 2$, $t = 8 \times 10^{-6}$), and ($L = 8$, $D = 3$, $t = 4 \times 10^{-6}$) . Within an avalanche interval, we define the variables included in the avalanche as those that changed sign and thus can affect the energy calculated from the orthant dynamics. A few of these configurations are shown in the main text and a more complete selection is displayed in Fig. 8.

Once a set of avalanches has been extracted from a simulation, we investigate the structure of the largest avalanche by calculating the probability that a spin lying a distance r away (measured in terms of lattice steps) from the first spin to flip is included within the avalanche. To this end, we define a cluster configuration as being $v_i = 1$ if the spin is included in the cluster and 0 otherwise. This acts as indicator variable which for an individual cluster allow us to calculate the probability that a spin a distance r away was flipped (recall that the lattices we generate are periodic and this distance is calculated as the minimum distance of a path between the two sites). This is then averaged across randomly generated instances and initial conditions of the solver.

The probability obtained may be interpreted as a correlation in the slow driving or instantonic limit in which the avalanche may be regarded as occurring at an instant in time, and calculated on the orthant dynamics of the system. As shown in the main text, the largest avalanche gives a finite probability for a spin anywhere in the lattice to change sign and is thus extensive.

V. SIMULATIONS AND SOLVER TUNING

With the exception of IBM CPlex and Falcon, solvers were implemented in Python 2.7 using the NumPy and SciPy libraries [41]. Simulated Annealing, Parallel Tempering, CPlex and the model in the main text were run at UCSD on a single core of an Intel Xeon E5430 with 16 Gb RAM.

A. Model Simulations

In order to limit the voltages and memristors to the allowed regions and make them robust to numerical errors, the equations simulated were,

$$\dot{v}_i = B_{v_i,(-1,1)}\Big(\sum_{\langle ij \rangle} J_{ij} x_{ij} v_j - (1 - x_{ij}) \frac{|J_{ij}|}{2} \big(v_i - \mathrm{sgn}(J_{ij}) v_j\big) \Big) \tag{15}$$

$$\dot{x}_{ij} = \beta B_{x_{ij},(0,1)}\Big(x_{ij}\big(1 - x_{ij}\big) \times \Big(\frac{|J_{ij}|}{2} \big(1 - \mathrm{sgn}(J_{ij}) v_i v_j\big) - \gamma \Big) \Big) \tag{16}$$

where $B_{x,(l,h)}(\cdot)$ implements the bounds to ensure integration steps that leave the region return to the fixed points as

$$B_{x,(l,u)}(f) = \begin{cases} (u - x), & x > u, f > 0 \\ (l - x), & x < l, f < 0 \\ f, & \text{otherwise.} \end{cases} \tag{17}$$

Integrations are carried out using forward Euler with parameters tuned for each set of instances to maintain the slow driving limit, hence allowing for an easy identification of avalanches. This tuning is *not* required to *solve* an instance, but it is in order to detect clearly defined avalanches. First, it was determined through tuning that $\beta = \frac{1}{400}$ with a maximum time of $t_{max} = 25,000$ gave well defined avalanches for $L = 15$, $D = 2$ and usually solved instances near to $t = t_{max}/2$. In order to maintain this limit for larger instances, the memristor timescale was

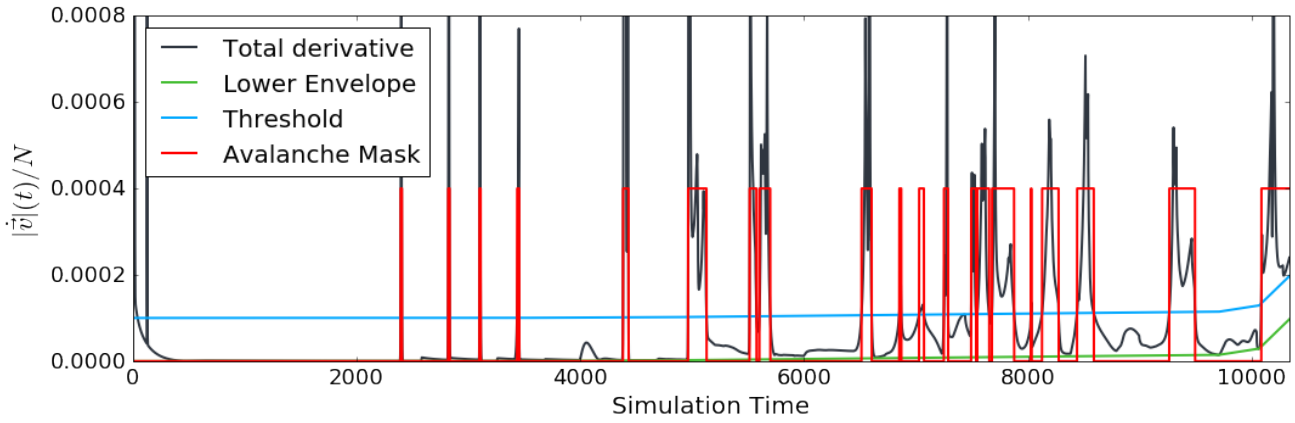

FIG. 7. **Extracting avalanches from a continuous trajectory.** Avalanches display as sharp spikes in the total voltage derivative vector of the system. Here, we have displayed the magnitude of the total voltage derivative of the system normalized to the number of spins N for the trajectory displayed in the main text ($L = 15$, $D = 2\,\alpha = 0.2$). As the timescale of the avalanches can slow, sometimes dramatically, over the course of the simulation, the convex lower bound of the derivative is first calculated. An avalanche interval is defined as a continuous period in which the system exceeds a threshold above this envelope (here chosen as 0.0001). Voltages that change sign during an interval are included in the avalanche configuration as shown in Fig. 8.

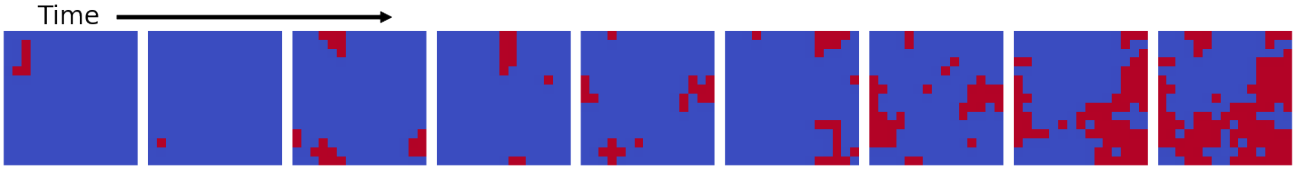

FIG. 8. **A sample of detected avalanches.** For the trajectory shown in the main text and the clusters detected in Fig. 7 a representative sample of the detected avalanche configurations are displayed, including the first and last avalanches in the trajectory. Over the course of the simulation, the average size of the avalanches grows until it reaches an extensive set of spins which can span the entire lattice.

slowed, scaling with the inverse square of the number of spins. Slowing the memristor timescale requires increasing the maximum simulation time in the same way, such that t_{max} was scaled with the square of the number of spins. For $L = 15$, $\gamma = 0.65$ was found to give a well defined transition, but as instance size increased this value would lead to quasi-periodic behavior more often and γ was increased to $L = 19$, $\gamma = 0.75$, $L = 23$, $\gamma = 0.85$ and $L = 8$, $D = 3$, $\gamma = 0.85$.

B. Simulated Annealing

Simulated annealing was implemented using a linear schedule in $\tilde{\beta}$ (the inverse temperature) from $\tilde{\beta}_i = 0.1$ to $\tilde{\beta}_f = 2$ such that at each step in the annealing a sweep of metropolis samples across the entire lattice was performed. For annealers to be most effective in a given computational time, the number of cooling steps (i.e., number of metropolis sweeps) performed on a single initial condition versus the number of initial conditions attempted must be optimized. As we are working with planted solution instances, runs consisted of continued repetitions at a fixed number of metropolis sweeps until the solution energy was encountered or some maximum allowed time was reached. Tuning was performed by running each instance with a varying number of temperature steps in the cooling schedule until the optimum was reached. This was performed on 600 frustrated loop instances ($d = 3$, $\alpha = 0.3$, $l \geq 6$) for $L = 6$ through 11 and the time to solution for each run was recorded. The results of these runs are plotted in Fig. 9 along with curves representing the $q = 0.99$ quantile.

Since the intention of this work is to examine the scaling for very large sizes, this analysis cannot be repeated for all sizes we intended to run. Instead, we used the values at small N to estimate the scaling of the optimal number of sweeps per repetition as shown by the curve in Fig. 9. Assuming a power law form for this dependence led to the scaling equation

$$\#sweeps_{opt} = (0.052)N^{1.326} \tag{18}$$

which was used to estimate the optimal number of sweeps for the scaling figure in the main text. Some effort was

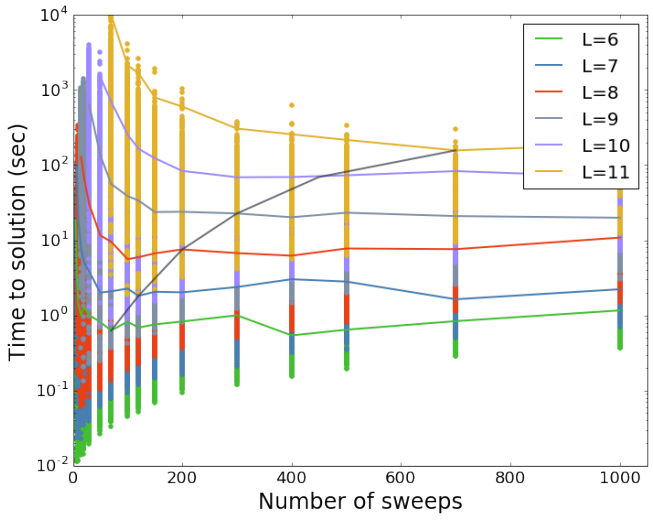

FIG. 9. **Tuning for simulated annealing.** Simulated annealing runs were performed on 600 3-D frustrated loop instances for $L = 6, 7, 8, 9, 10, 11$ at varying number of metropolis sweeps. As can be seen, if the number of sweeps is too few the ground state is only rarely encountered. Beyond a certain amount, more sweeps will have a negligible or slightly negative effect on the runtime. We estimated the location and scaling of this crossover with the dark curve, allowing us to extend the runs of this algorithm to larger sizes.

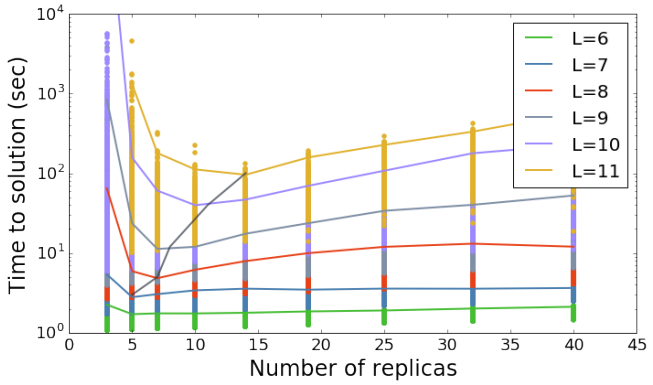

FIG. 10. **Tuning for parallel tempering.** Parallel tempering runs were performed on 600 frustrated loop instances for $L = 6, 7, 8, 9, 10, 11$ with varying numbers of replicas. For sufficiently large sizes ($L > 6$) a clear optimum was observed in the time to solution, with the number of replicas at the optimum growing with the number of spins. The location of these optima were used to estimate a scaling law for the optimal number of replicas at larger sizes.

made to slightly overestimate the scaling, since a modest overestimate has a less dramatically negative effect than an underestimate.

C. Parallel Tempering

The parallel tempering (PT) algorithm we employed used a ladder of temperatures linearly spaced in $\tilde{\beta}$ from $\tilde{\beta}_h = 3$ to $\tilde{\beta}_l = 0.5$ which were found to produce solutions in the fastest time. One step of the algorithm consisted of a single sweep of metropolis sampling over all replicas, followed by a single proposed exchange where replicas at neighboring temperatures had their configurations switched according to the probability,

$$P_{exchange} = \min\{1, \exp\Big((\tilde{\beta} - \tilde{\beta}')(E - E')\Big)\}, \qquad (19)$$

where E and E' are energies of the instances. This cycle of metropolis sweeps and exchanges was repeated until the solution was reached. As the system size increases, the extensivity of the energy causes exchanges to become more rare and so the number of temperatures simulated should be increased with the size of the system. Therefore, in order for this implementation to run efficiently, simulations must be run with a number of temperatures that balances the diffusion of configurations through the temperatures with the computational cost of metropolis sweeps over the larger number of systems. In order to estimate this, the time to solution was found for 600 frustrated loop instances (dimensions $D = 3$, density $\alpha = 0.3$, loop length $l \geq 6$) for $L = 6$ through 11 as shown in Fig. 10, along with a curve representing the $q = 0.99$ quantile. As with simulated annealing, a scaling law for the optimal number of replicas was extracted and found to follow

$$\#replicas_{opt} = (0.066)N^{0.74}. \qquad (20)$$

This allows the algorithm to transfer some computation time to memory which ultimately became the limiting factor in our PT simulations. While we could constrain the memory usage of the algorithm, this would likely increase the computation time, so the scaling we see in Fig. 4 of the main text may be regarded as a conservative estimate.

D. CPlex

CPlex was run using the python API within the IBM ILOG CPLEX Optimization Studio version 12.7.1.0 under an academic license [27]. The QUBO (quadratic unconstrained binary optimization) form for the associated frustrated loop instance was found through the transformation to binary variables, $s_i = 2b_i - 1$ which leads to the correspondence,

$$E_{SG} = -\tfrac{1}{2}\textstyle\sum_{ij} J_{ij} s_i s_j \qquad (21)$$

$$= \tfrac{1}{2}\textstyle\sum_{ij} Q_{ij} b_i b_j + C \qquad (22)$$

$$Q_{ij} = \begin{cases} -4J_{ij}, & i \neq j \\ 4\sum_j J_{ij}, & i = j \end{cases} \qquad (23)$$

$$C = -4\textstyle\sum_{ij} J_{ij}. \qquad (24)$$

Within CPlex, problems in this form are first transformed to a mixed integer programming (MIP) form. Unlike the other solvers in this work, CPlex is a complete solver and will attempt a proof of optimality along with solving the instance. To prevent this, a callback was employed that terminates the search once the planted solution energy was found. Cuts were set to balance optimality and feasibility in the search.

E. Falcon

The memcomputing solver, Falcon, was implemented with MATLAB as specified in [21] as a Boolean satisfiability solver which accepts instances in conjunctive normal form [34]. The transformation to CNF shown in section I has been performed on the frustrated loop instances and simulations were carried out on a single core of an Intel Xeon 6148 with 192 GB RAM. The Falcon parameters were first tuned for the smallest size instances, and then the same parameters have been employed to solve all instances reported in Fig. 4 of the main text. Since Falcon integrates differential equations numerically (using forward Euler), it employs memory that scales linearly with problem size [42].

VI. FITTING

Fitting of each solver was performed with the SciPy optimization library's curve fit function [41], which utilizes non-linear least squares to fit a particular function. The time to solution for instances was found to be approximately log-normally distributed, as may be observed from the consistent standard deviations (0.5-1.5 orders of magnitude) found in log-space in Fig. 11. Fitting was thus performed in log-space to estimate the worst case time to solution from the sample maximum of the instance class. For each solver, fits were performed for a polynomial, $TTS \approx AN^B$ and exponential $TTS \approx ae^{bN^c}$ trend as is displayed in Fig. 11. For Falcon, an exponential fit did not converge and so only a polynomial fit is shown following $TTS \approx (3.4 \times 10^{-8}\, sec)N^{2.3}$.

For the other solvers, both a polynomial and exponential fit was found to converge and so more care has been taken in selecting the appropriate trend. Parallel tempering and CPlex appear to clearly favor the exponential fit, with parallel tempering following $TTS \approx (0.0051\, sec)e^{0.90N^{0.33}}$ over the polynomial fit $TTS \approx (9.7 \times 10^{-9}\, sec)N^{3.2}$, and CPlex favoring $TTS \approx (2.4 \times 10^{-5}\, sec)e^{4.5N^{0.16}}$ over $TTS \approx (3.9 \times 10^{-7}\, sec)N^{2.6}$.

In the case of SA, the polynomial fit is found to follow, $TTS \approx (5.0 \times 10^{-10}\, sec)N^{3.6}$, while the exponential fit is $TTS \approx (1.3 \times 10^{-10}\, sec)e^{11.N^{0.13}}$. Although the two fits may appear close, we note that SA (as well as PT and CPlex) could not converge for all instances within the allotted time of one week. This suggests that the fits may underestimate their actual scaling. In addition, in [15], frustrated loop instances were tested on the Chimera graph and found to scale exponentially for a variety of local search algorithms. We thus expect a similar dependence here as well.

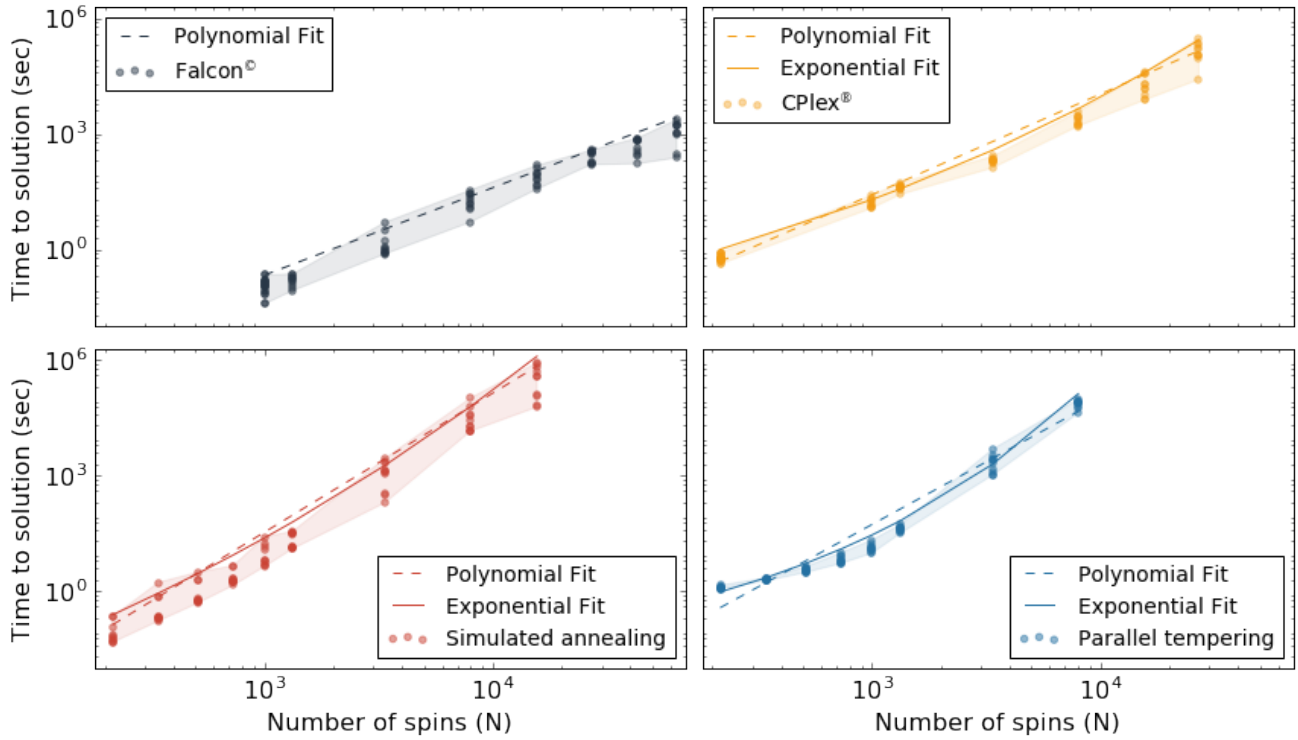

FIG. 11. **Fitting the Time to Solution.** For each solver, the geometric average complexity of the instance class was estimated by fitting the time to solution data for a set of 10 frustrated loop spin glasses ranging in size across several orders of magnitude. Polynomial ($TTS \approx AN^B$) and exponential ($TTS \approx ae^{bN^c}$) trends were fit to the data to assess the performance of each approach. For Falcon, only the polynomial fit was found to converge and it demonstrated the lowest exponent of all fits ($B = 2.3$). All other solvers were found to scale exponentially although SA appears close to sub-exponential. However, note that CPlex, SA and PT could not converge for all instances within the allotted time of one week. Hence, the corresponding fits may underestimate their actual scaling.