UNIVERSITY OF CALIFORNIA SAN DIEGO

Investigation of the Physics of Digital Memcomputing Machines

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Physics

by

Sean Rhett-Burke Bearden

Committee in charge:

Professor Massimiliano Di Ventra, Chair
Professor Gert Cauwenberghs
Professor Ivan K. Schuller
Professor Tatyana Sharpee
Professor Bradley T. Werner

2020

The dissertation of Sean Rhett-Burke Bearden is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California San Diego

2020

DEDICATION

To my mother and father

EPIGRAPH

Problems always appear to be intractable

until we discover efficient algorithms for solving them.

—Garey and Johnson

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thank you to Dr. Massimiliano Di Ventra for his guidance and tutelage. Without his invitation to join his research group I have no idea where I would be today.

Thank you to Dr. Igor Žutić for offering me a research opportunity early in my undergraduate studies. His mentorship in research gave me confidence in my abilities.

Thank you to Sharmila Poddar for dealing with my frequent stops by her office. She is the unsung heroine of the Physics Department.

Thank you to the numerous mentors I've encountered during my time at UC San Diego, including, Dr. Dmitri Basov, Dr. Fabio L. Traversa, Dr. John M. York, Dr. Vish Krishnan, Sensei Phillip Palmajar, Shana Slebioda, Sinai Cota, Dragos Craciun, and Richard Mayhew.

Chapter 2, in full, is a reprint of the material as it appears in Instantons in Self-Organizing Logic Gates in Physical Review Applied, 2018, Bearden, Sean R.B.; Manukian, Haik; Traversa, Fabio L.; Di Ventra, Massimiliano, American Physical Society, 2018. The dissertation author was the primary researcher and author of this publication.

Chapter 4, in full, is a reprint of the material as it appears in Critical Branching Processes in Digital Memcomputing Machines in Europhysics Letters, 2019, Bearden, Sean R.B.; Sheldon, Forrest C.; Di Ventra, Massimiliano, IOP Publishing, 2019. The dissertation author was the primary researcher and author of this publication.

Chapter 5 and Appendix A, in full, have been submitted for publication. Bearden, Sean R.B.; Pei, Yan Ru.; Di Ventra, Massimiliano, 2020. The dissertation author is the primary researcher and author of the main text (Ch. 5), and Yan Ru Pei is the primary researcher and author of the supplementary information (Appx. A).

Appendix B, in part, is a reprint of the material as it appear in the supplementary information of Mode-Assisted Unsupervised Learning of Restricted Boltzmann Machines in Communications Physics, 2020, Manukian, Haik; Pei, Yan Ru; Bearden, Sean R.B.; Di Ventra, Massimiliano, Nature Publishing Group, 2020. The dissertation author is a co-author of this publication.

VITA

2011	A. S. in Natural Sciences, Ohio University, Athens
2014	Barry Goldwater Scholarship
2015	B. S. in Physics and Applied Mathematics *summa cum laude*, State University of New York at Buffalo
2015	National Science Foundation Graduate Research Fellowship
2015	Alfred P. Sloan University Center of Exemplary Mentoring Scholarship
2016-2020	Graduate Student Researcher, Massimiliano Di Ventra Group, University of California San Diego
2017	M. S. in Physics, University of California San Diego
2020	University of California President's Dissertation Year Fellowship
2020	Ph. D. in Physics, University of California San Diego

PUBLICATIONS

Bearden, Pei, Di Ventra, "Efficient Solution of Boolean Satisfiability Problems with Digital Memcomputing," *submitted*, 2020.

Manukian, Pei, Bearden, Di Ventra, "Mode-assisted unsupervised learning of restricted Boltzmann machines," *Communications Physics*, 3(1):1-8, 2020.

Bearden, Sheldon, Di Ventra, "Critical branching processes in digital memcomputing machines," *EPL (Europhysics Letters)*, 127(3):30005, 2019.

Bearden, Manukian, Traversa, Di Ventra, "Instantons in self-organizing logic gates," *Physical Review Applied*, 9:034029, 2018.

ABSTRACT OF THE DISSERTATION

Investigation of the Physics of Digital Memcomputing Machines

by

Sean Rhett-Burke Bearden

Doctor of Philosophy in Physics

University of California San Diego, 2020

Professor Massimiliano Di Ventra, Chair

This dissertation will review and compile several advancements in the development of digital memcomputing machines. Memcomputing is an efficient computing paradigm that uses memory to process and store information in the same physical location. Digital memcomputing machines have been introduced as a scalable version of the memcomputing paradigm. The memcomputing paradigm can be used to solve difficult constraint satisfaction and combinatorial optimization problems. Herein, Boolean satisfiability problems will be used as benchmarks. This dissertation will introduce the digital memcomputing machine, detailing self-organizing logic circuits and investigating the operation of their fundamental units: self-organizing logic gates. The dynamical system that describes a digital memcomputing machine will be numerically

integrated with a forward-Euler scheme. The robustness of the dynamical system to noise allows for the use of a simple integration scheme. We find power-law scalability in the typical-case of hard clause distribution control instances of 3-SAT. We anticipate our results to broaden research directions in physics-inspired computing paradigms ranging from theory to application, from simulation to hardware implementation.

Chapter 1

Digital Memcomputing Machines

Prior to beginning my research of digital memcomputing machines, Drs. Massimiliano Di Ventra, Yuriy V. Pershin, and Fabio L. Traversa, along with others, had made many advances studying memcomputing machines [DP13b, TD15, TD17]. I will detail the background of memcomputing so as to put my contributions in context, but will focus on the digital versions of memcomputing, rather than analog versions. See Ref. [DT18] for a self-contained account of scalable digital memcomputing machines.

1.1 Background

Herein, there will be many references to modern computers and the computation they perform. It is important to realize that computing is fundamentally a physical process [DT18]. The statement may seem obvious when considering the physical processes harnessed by the electronic components of computers (for example, transistors), however, virtually any physical process can be harnessed for some form of computation. Note, that we are speaking of Alan Turing's model of computation [GJ90], that is, a mapping (transition function) between two sets of finite symbols (input and output) in discrete time.

It is important to distinguish between continuous and discrete time. Memcomputing

machines operate in continuous time[1], though, their simulations on modern computers require the discretization of time. Continuous time is *physical time*: a fundamental physical quantity. Discrete time is not a physical quantity, and might be best understood as *counting time*: counting something (function calls, integration steps, etc.) to give an indication (perhaps approximation) of the physical time. In the literature of Physics and other Physical Sciences, physical time has an assigned SI unit of seconds, whereas in Computer Science and related disciplines, counting time is dimensionless.

Granting the infinite resources utilized by a Turing machine[2], universal memcomputing machines (UMMs) have been shown to be Turing-complete, meaning UMMs can simulate universal Turing machine (UTM) [TD15]. The UMM class contains digital and analog machines. While analog memcomputing machines theoretically have tremendous computational power, analog systems cannot be engineered for scalability, as their growing size requires growing resources to achieve the same accuracy [DT18]. It is the digital memcomputing machine (DMM) that is scalable, and the focus of this dissertation.

The informed reader may notice the P vs. NP problem lingering in the background of this dissertation. There is no statement on the P vs. NP problem implied by the work herein. However, the reader should be aware the conjecture P $\neq$ NP is often assumed for algorithms that are inherently evaluated in discrete time. While the simulations of DMMs herein are technically algorithms due to the necessity of converting continuous time to discrete time, a physical DMM would necessarily operate in continuous time. Thus, the framework by which the P vs. NP problem is formulated cannot be applied to a physical DMM (not a simulation), as discretization is not applicable to continuous (physical) time.

[1] A physical process is necessarily continuous in time, as discrete time is not physical, rather a necessary consequence of simulating a physical system.

[2] A physical realization of memcomputing will, of course, have finite resources. However, the "machines" used to study computational complexity are theoretical and impossible to build

1.2 Circuits of Logic Gates

Modern computers rely on the implementation of *uni-directional* logic gates that represent Boolean functions [Par10]. Circuits built to simulate Boolean functions are desirable because they are deterministic: A unique input has a unique, reproducible output.

Modern computers relegate the task of logic to central processing units (CPUs). However, the resources required for the task might exhaust the resources present within the CPU, specifically, cache memory. For typical processes on modern computers, random-access memory (RAM) is the memory used for data and machine code, and is external to the CPU. The physical separation of CPU and RAM results in what is known as the *von Neumann bottleneck*, a slow down in computation caused by the transfer of information between physical locations [Bac78].

The circuits herein will be built from self-organizing OR, AND, and NOT gates. For completeness, we review the truth tables associated with OR and AND gates with three input terminals in Table 1.1. A NOT gate has one input and one output, and simply negates the logical variable on the input to return on the output.

Table 1.1: Standard truth tables for three-input AND and OR Boolean operators. The AND operator returns TRUE (T) if and only if all inputs are TRUE. Conversely, the OR operator returns FALSE (F) if and only if all inputs are FALSE.

AND			
In_1	In_2	In_3	Out
F	F	F	F
T	F	F	F
F	T	F	F
F	F	T	F
T	T	F	F
T	F	T	F
F	T	T	F
T	T	T	T

OR			
In_1	In_2	In_3	Out
F	F	F	F
T	F	F	T
F	T	F	T
F	F	T	T
T	T	F	T
T	F	T	T
F	T	T	T
T	T	T	T

1.3 Self-Organizing Logic Circuits

To overcome the von Neumann bottleneck, Drs. Di Ventra and Pershin proposed computing with and in memory, naming the approach memcomputing[3] [DP13b]. Distinct from in-memory computation [ZCO+15], it is an efficient computing paradigm that uses memory to process and store information in the same physical location.

A digital memcomputing machine is realized as a self-organizing logic circuit (SOLC). These circuits differ from traditional logic circuits in that input and output terminals are no longer distinct. In a traditional logic circuit, some input is given and the output is the result of computation performed on the input, via uni-directional logic gates. In contrast, a SOLC can be operated by assigning the output terminals, then reading the input terminals.

Operating a logic circuit "backwards" has many applications. An example is integer factorization: Given an integer, factor it into its prime factors. For simplicity, assume the given integer, b, is the product of two prime numbers, p and q. If given p and q, then a multiplication circuit can be employed to find the product, b, of the two prime numbers. A traditional logic circuit, appropriately designed, can easily perform this task. Now, if given b and told it can be factored into two prime numbers, we take the same multiplication circuit structure (logic gates connected similarly), but design it to be a SOLC so the logic gates become *terminal agnostic*, meaning signal can be received and sent on any terminal of the logic gate. However, the new logic gates are not bijective (see Table 1.1), so the entire circuit will have to self-organize to produce the values of p and q on the "input" terminals. These SOLCs are built using self-organizing logic gates.

[3] Where "mem" is short for memory.

1.4 Self-Organizing Logic Gates

To understand the operation of a SOLC, we must investigate the behavior of its fundamental unit, the self-organizing logic gate (SOLG). For simplicity, we will narrow our focus to OR and AND gates. (There is no need for a self-organizing NOT gate, as it is trivially implemented.) A SOLG is a *bi-directional* logic gate, meaning input and output terminals can both send and receive signals. It is this unique property of SOLGs that allows for a SOLC to self-organize into equilibrium, indicating a solution has been found.

1.5 Remarks

DMMs were shown to employ instantons (see Ch. 2) in self-organizing logic *circuits* [DTO17]. My first investigation involved determining whether the instantons were employed at the level of the self-organizing logic *gate*, or if the circuit must be constructed before the instantons emerge. This research is reproduced in the next chapter.

Chapter 2

Instantons in Self-Organizing Logic Gates

Self-organizing logic is a recently-suggested framework that allows the solution of Boolean truth tables "in reverse," i.e., it is able to satisfy the logical proposition of gates *regardless* to which terminal(s) the truth value is assigned ("terminal-agnostic logic"). It can be realized if time non-locality (memory) is present. A practical realization of self-organizing logic gates (SOLGs) can be done by combining circuit elements with and without memory. By employing one such realization, we show, numerically, that SOLGs exploit elementary instantons to reach equilibrium points. Instantons are classical trajectories of the non-linear equations of motion describing SOLGs, and connect topologically distinct critical points in the phase space. By linear analysis at those points, we show that these instantons connect the initial critical point of the dynamics, with at least one unstable direction, directly to the final fixed point. We also show that the memory content of these gates only affects the relaxation time to reach the logically consistent solution. Finally, we demonstrate, by solving the corresponding stochastic differential equations, that since instantons connect critical points, noise and perturbations may change the instanton trajectory in the phase space, but not the initial and final critical points. Therefore, even for extremely large noise levels, the gates self-organize to the correct solution. Our work provides a physical understanding of, and can serve as an inspiration for, new models of bi-directional

logic gates that are emerging as important tools in physics-inspired, unconventional computing.

2.1 Introduction

Traditional Boolean logic is uni-directional, namely, given the truth value of a set of input terminals, one finds the consistent output value according to a given truth table [BC07]. This is the type of logic that is employed, e.g., in our standard computing paradigm [Par10].

Recently, a new type of logic has been introduced by two of us (FL and MD) [TD17] that is both "invertible" and "terminal-agnostic." This means that, in addition to working as traditional Boolean logic does from input terminals to output terminals, it can work "in reverse," without reference to any particular set of terminals: by assigning a truth value to *any* terminal (even those at the traditional output), the gate is able to find a *logically consistent* truth assignment of the other terminals [TD17]. Of course, this logic is not necessarily bijective, because, in most cases, logic gates have a different number of terminals on one end of the gate than the other.

The physical ingredient to realize such a framework is *time non-locality* (memory) [TD17]. Memory allows the system to *self-organize* into the correct truth value according to the initial conditions assigned [TD15]. For this reason, these gates were named *self-organizing logic gates* (SOLGs) [TD17, MTD17].

Note that the self-organizing logic we consider here has no relation to the invertible universal Toffoli gate that is employed, e.g., in quantum computation [Tof80]. Toffoli gates are truly one-to-one invertible, having 3-bit inputs and 3-bit outputs. On the other hand, SOLGs need only to satisfy the correct logical proposition, without a one-to-one relation between any number of input and output terminals. Instead, it is worth mentioning another type of bi-directional logic that has been recently discussed in Ref. [CFSD17] using stochastic units (called p-bits). These units fluctuate among all possible consistent inputs. However, in contrast to that work, the invertible logic we consider here is *deterministic*.

With time being a fundamental ingredient, a dynamical systems approach is most natural to describe such gates. In particular, *non-linear* electronic (non-quantum) circuit elements with and without memory have been suggested as building blocks to realize SOLGs in practice [TD17] (see also Fig. 2.1).

By assembling SOLGs with the appropriate architecture, one then obtains circuits that can solve complex problems efficiently by mapping the equilibrium (fixed) points of such circuits to the solution of the problem at hand, as shown in, e.g., Refs. [TD17, MTD17, MTD19, DTO17]. Moreover, it has been proved that, if those systems are engineered to be point dissipative [Hal10], then, if equilibrium points are present, they do not show chaotic behavior [DT17a] or periodic orbits [DT17b].

It was subsequently demonstrated [DTO17], using topological field theory (TFT) applied to dynamical systems, that these circuits are described by a Witten-type TFT [Wit88], and they support long-range order, mediated by *instantons*. Instantons are classical trajectories of the non-linear equations of motion describing these circuits (see, e.g., [Col77] or [HKK$^+$00]).

Instantons have been introduced first in the field of high-energy Physics to compute more efficiently tunneling matrix element between local vacua by a Wick rotation to Euclidean space (see, e.g., [SS98]). Local vacua are then transformed into critical points of the corresponding classical equations of motion [Col77]. Therefore, instantons can be viewed as the classical analog of "tunneling" in the phase space. In fact, instantons can only connect critical points with different indices, namely different number of unstable directions. In turn, critical points can be located anywhere in the phase space. Therefore, instantons can be highly non-local objects. Finally, since critical points are related to the topology of the phase space, their number and index are robust against noise and perturbations [Fom12]. In other words, one needs to break the topology of the phase space to change its critical points. In practice, this requires changing the physical system itself.

The intrinsic non-locality of instantons, coupled with the topological character of critical

points, is reminiscent of the "rigidity" and topological character of the ground state of some strongly-correlated quantum systems that are currently investigated for topological quantum computation, namely quantum computation that is robust against dephasing and noise [FKLW03, NSS$^+$08, Kit03]. This analogy is not far-fetched. In fact, in the case of self-organizing circuits, instantons, by connecting topologically-distinct critical points in the phase space, correlate elements of the circuit non-locally in space and time [DTO17]. This non-locality is somewhat reminiscent of quantum entanglement. However, SOLGs are circuits that achieve long-range order without quantum-mechanical effects.

The long-range order is not surprising since TFTs with condensed instantons are known to be log-conformal, hence support gapless excitations [FLN07]. Our previous work, however, leaves open the question as to whether the *single* SOLGs employ instantons as well, and, if so, what is the nature of the corresponding critical points.

In this paper, we answer these questions by numerically solving the differential equations of self-organizing AND (SO-AND) and OR (SO-OR) gates *both with and without noise*. The set of Boolean operators $\{$AND, NOT$\}$ forms a functionally complete set, i.e., the two gates form a basis for all Boolean logic, as does the set $\{$OR, NOT$\}$. The NOT gate is implemented trivially in an electronic circuit, since it is simply a current (or voltage) inverter [Nea01], and, therefore, it is not described herein.

We find that the dynamics of these self-organizing gates proceeds as follows. Given an arbitrary initial condition the system "scatters" into unstable critical points whose unstable direction has an eigenvalue of the Jacobian matrix which is, in absolute value, considerably smaller than the largest eigenvalue of the stable directions. This makes them almost attractive to the initial dynamics. Subsequently, an instanton connects the unstable critical point to the equilibrium (fixed) point. In addition, the unstable direction evolves into a center manifold of the final fixed point. We also explicitly show, by perturbing the initial conditions and by solving stochastic differential equations, that although the trajectories connecting critical points

may be substantially different due to either perturbations or noise, the instantons *always* go to the (unchanged) final fixed points. Using a simple diffusion model for typical memristors made of oxides, we relate the noise intensity to temperature. We find that even at very large temperatures (beyond the stability of the underlying materials) the SOLGs keep operating as expected. Therefore, the single logic units of more complicated self-organizing circuits take advantage of the instantonic long-range order, thus allowing the system to explore a vast phase space very efficiently, even in the presence of noise.

These results suggest that the topological features of SOLGs are essential in their operation as units of computation. In addition, our findings may provide a physical understanding of other types of recently suggested (stochastic) bi-directional logic gates employed in unconventional computing [CFSD17, PSJ$^+$17].

2.2 SOLGs formulation

Let us start by outlining a model of SOLGs as a system of coupled, nonlinear, ordinary differential equations. We will then solve these equations numerically to identify instantons, their critical points, and, by diagonalizing the Jacobian (the matrix of the derivative of the flow vector field), their topological features.

The implementation of SOLGs using electronic circuits is not unique, provided some of their properties are preserved [TD17, DTO17]. We refer the reader to Ref. [TD17] for all the mathematical properties of these gates. In order to make the phase space as small as possible – hence the numerical analysis as easy as possible – we choose a much simpler representation of SOLGs than that proposed in Ref. [TD17], which accomplishes the same tasks with a fewer number of variables [1].

[1] Note, however, that in this very simplified representation of SOLGs, there may be stable critical points that do not satisfy Boolean logic. These cases are easily removed by adding voltage-controlled differential-current generators as in Ref. [TD17]. However, the subsequent increased dimensionality of the phase space would render the numerical analysis necessarily more complex.

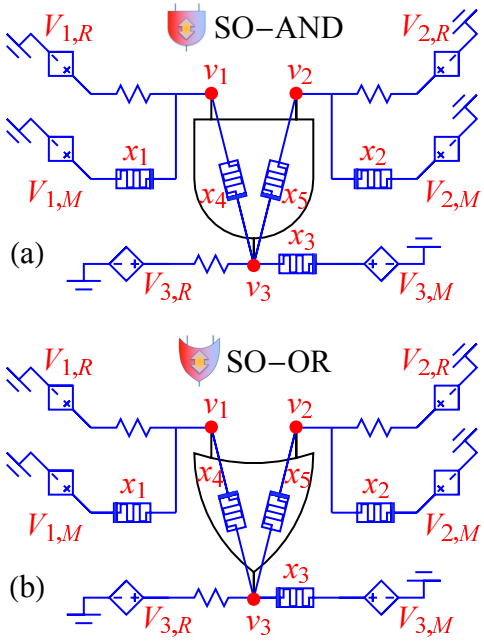

Figure 2.1: Circuit diagrams representing (a) a self-organizing AND (SO-AND) gate, and (b) a self-organizing OR (SO-OR) gate. The memristive elements (represented by a rectangle with a square waveform inside) have state variables x_j, $j = 1,\ldots,5$; an orientation, denoted by the bar on one side; and contain a parasitic capacitor in parallel. All resistors have the same resistance value. The form of the voltage-controlled voltage generators (represented by a diamond shape with $+$ and $-$ signs) is given in Table 2.1. Voltages at the terminals are represented with a subscript M if they are at a memristive terminal, with a subscript R if they are at a resistance terminal.

In Fig. 2.1, we show the SO-AND/OR gates we employ in this work. They are modeled with standard resistors, resistors with memory (memristive elements) [DPC09], and voltage-controlled voltage generators (VCVGs) [TD17]. The memristive elements contain a capacitance in parallel, representing parasitic capacitive effects. The difference between the circuitry of the SO-AND and SO-OR gates is the orientation of the memristive elements, and the definitions of the VCVGs (see Table 2.1).

We want these gates to self-organize into the correct logical proposition irrespective of the terminal to which the truth value is assigned. To better understand how this is accomplished,

Table 2.1: Coefficients for the voltage-controlled voltage generators' relations given by $V_{i,j} = b_1 v_1 + b_2 v_2 + b_3 v_3 + dc_{gate}$, where $i = 1,2,3$ and $j = R,M$. All voltage are in Volts.

	b_1	b_2	b_3	dc_{AND}	dc_{OR}
$V_{1,M}$	0	-1	1	1	-1
$V_{1,R}$	3	1	-2	-1	1
$V_{2,M}$	-1	0	1	1	-1
$V_{2,R}$	1	3	-2	-1	1
$V_{3,M}$	2	2	-1	-2	2
$V_{3,R}$	-3	-3	5	2	-2

it is beneficial to start from a specific example. Let us then choose to encode the logical 1 (True) with 1 V and the logical 0 (False) with -1 V.

Consider first the SO-AND. If we set the voltage v_1 to 1 V, the system should evolve to either $v_2 = v_3 = 1$ V or $v_2 = v_3 = -1$ V. Both are logically consistent with an AND truth table. On the other hand, if we consider the SO-OR gate, and fix v_1 to -1 V (logical 0), the system should evolve to either $v_2 = v_3 = -1$ V or $v_2 = v_3 = 1$ V. The final result will depend on the initial conditions, namely the initial values of all voltages and internal state variables.

Below, we describe a set of dynamical equations that accomplishes the above tasks. For the evolution of the memristive state variables we choose an equation of motion of the form [TD17],

$$\frac{d}{dt}x_j = -\alpha h(x_j, v_{M_j}) g(x_j) v_{M_j}, \tag{2.1}$$

where x_j is the state variable for the j-th memristive element. The function h serves to cutoff the dynamics of the state variable in certain regimes. We choose the conductance of these elements, $g(x) = ((R_{off} - R_{on})x + R_{on})^{-1}$, where we set $R_{off} = 1\ \Omega$ and $R_{on} = 0.01\ \Omega$. Thus, $g(x)v_M$ is equal to the current flowing through a memristor. The voltage drop, v_M, is measured based on the orientation of the memristor: $v_M = v_a - v_b$, where v_b is measured from the thick-bar side of the electronic symbol for the memristor. The coefficient α is restricted to be positive, and we choose $\alpha = 60$. The physical meaning of α is discussed in Ref. [DP13a]. Finally, the values of the state variables are bounded, and are typically chosen to be $x \in [0,1]$ [TD17].

Ideally, in order to strictly enforce $x \in [0,1]$, $h(x, v_M)$ should be represented by step functions [TD17]. However, in practical realizations and numerical simulations, the step functions should be replaced by some differentiable function. We use, [TD17]

$$\begin{aligned} h(x, v_M) = (1 - e^{-kx})\hat{\theta}^r \left(\frac{v_M}{2V_t} \right) + \\ (1 - e^{-k(1-x)})\hat{\theta}^r \left(-\frac{v_M}{2V_t} \right), \end{aligned} \tag{2.2}$$

where $k = 2$, and choose $V_t = 0.1$ V. The $\hat{\theta}^r$ function is defined as,

$$\hat{\theta}^r(y) = \begin{cases} 1 & y > 1 \\ \sum_{i=r+1}^{2r+1} a_i y^i & 0 \leq y \leq 1 \\ 0 & y < 0 \end{cases} \tag{2.3}$$

where we use the simplest case, $r = 1$. The coefficients can be found by requiring continuity and differentiability in $y = 0$ and $y = 1$. This is equivalent to satisfying equations $\sum_{i=r+1}^{2r+1} a_i = 1$ and $\sum_{i=r+1}^{2r+1} \binom{i}{l} a_i = 0$ for $l = 1, \ldots, r$. The coefficients for our implementation are $a_2 = 3$ and $a_3 = -2$.

If we analyze the particular case discussed above, we fix, for both SOLGs, the voltage generator on terminal 1, and we perform standard nodal analysis on terminals 2 and 3 to find (see also Fig. 2.1),

$$\begin{aligned} C(-\frac{d}{dt}v_1 - 2\frac{d}{dt}v_2 + 2\frac{d}{dt}v_3) = -i_2 + \\ (v_2 - v_3)g(x_5) + (-V_{2,M} + v_2)g(x_2) + \frac{-V_{2,R} + v_2}{R}, \end{aligned} \tag{2.4}$$

$$\begin{aligned} C(-3\frac{d}{dt}v_1 - 3\frac{d}{dt}v_2 + 4\frac{d}{dt}v_3) = -i_3 + (v_1 - v_3)g(x_4) + \\ \frac{V_{3,R} - v_3}{R} + (v_2 - v_3)g(x_5) + (-v_3 + V_{3,M})g(x_3), \end{aligned} \tag{2.5}$$

where the capacitance is $C = 10^{-5}$ F and $R = 1$ Ω [2]. The VCVGs generate a voltage from

[2] In this example, we have used a large parasitic capacitance. However, this is not necessary as shown in

the relation $V_{i,j} = b_1 v_1 + b_2 v_2 + b_3 v_3 + dc_{gate}$, with dc_{gate} a constant voltage specific to each gate [TD17]. The coefficients, b_k, along with dc_{gate}, are given in Table 2.1. Terminals 2 and 3 are floating, therefore, $i_2 = i_3 = 0$. Additionally, $\frac{d}{dt} v_1 = 0$, due to terminal 1 being attached to a voltage generator that is held constant. The role of the VCVGs is to inject a large current when the gate is in an inconsistent configuration, a small current otherwise.

By solving numerically Eqs. (2.1), (2.4), and (2.5), with appropriate substitutions, we obtain precisely what we were after: a consistent logical solution for the given gate. This is reported in Fig. 2.2, where, for the particular initial conditions chosen, we obtain a consistent solution for each SOLG: for the SO-AND, by starting at the logical 1 for v_1, we obtain the logical 1 at both v_2 and v_3. Instead, for the SO-OR, by starting at the logical 0 at v_1, we obtain the logical 0 at both v_2 and v_3.

In the general case, the evolution of the terminal voltages and the memristive state variables of the SOLGs can be written compactly as,

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)), \tag{2.6}$$

where $\mathbf{x} = \{v_1, \ldots, v_m, x_1, \ldots, x_n\} \in X$ (X is the phase space) represents the voltages, v_j, the internal state variables of the memristors, x_j, and $\mathbf{F}$ is a system of nonlinear ordinary differential equations, representing the flow vector field.

The dynamical variables of the system then inhabit a phase space, $X \subset \mathbb{R}^{m+n}$. For the SO-AND/OR gates, $m = 3$ and $n = 5$. For the numerical simulations shown in Fig. 2.2 we have chosen to hold v_1 constant, so that the system has only seven dynamical variables.

Ref. [TD17]. A large capacitance value simply causes the peak of the instantonic trajectory to broaden, hence simplifying further the numerical analysis.

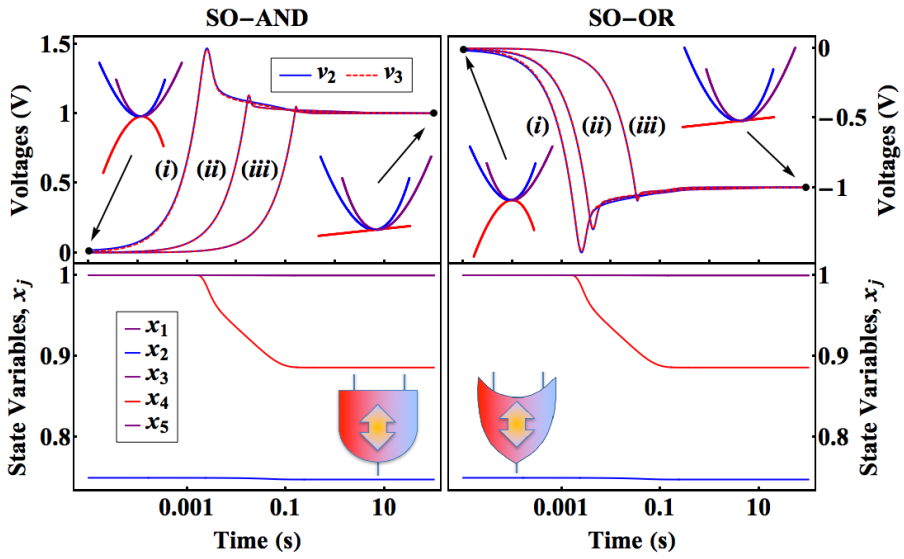

Figure 2.2: Time evolution of the voltages (top-left panel) and state variables (bottom-left panel) of the SO-AND compared to the time evolution of the voltages (top-right panel) and state variables (bottom-right panel) of the SO-OR. Elementary instantons in the SO-AND gate (top-left) are shown for (*i*) $\delta v_2 = 10^{-2}V$, (*ii*) $\delta v_2 = 10^{-3}V$, (*iii*) $\delta v_2 = 10^{-4}V$; Elementary instantons in the SO-OR gate (top-right) are shown for (*i*) $\delta v_2 = 10^{-2}V$, (*ii*) $\delta v_2 = 5\times 10^{-3}V$, (*iii*) $\delta v_2 = 5\times 10^{-4}V$. For perturbation $\delta v_2 = 10^{-2}V$, the particular choice of initial conditions results in the memristor state variables evolving identically for both gates: (bottom-left) SO-AND; (bottom-right) SO-OR. Only memristors associated with x_2 and x_4 evolve in time. In addition, the voltage evolution of the SO-AND and the SO-OR are specularly symmetric (observe the $\delta v_2 = 10^{-2}V$ case), as expected by their truth tables. The instanton connects the initial time critical point with two stable directions (positive curvature parabolas) and one unstable direction (negative curvature parabola), with the final state critical point with all three stable directions (fixed point). The unstable direction evolves into a center manifold (flat red line).

2.3 Instantons and stability analysis

Solutions $\mathbf{x}_{cr}$ to $\mathbf{F}(\mathbf{x}_{cr}) = 0$ are the critical points in the phase space we are after. We have performed an extensive search of critical points of Eq. 2.6 in the phase space, and found some with one unstable direction, and some with two unstable directions. Since our goal is simply to show that instantons are present in SOLGs, we focus on those originating from initial critical points with only one unstable direction.

One such critical point is $\mathbf{x}_{cr} = \{v_2, v_3, x_1, x_2, x_3, x_4, x_5\} = \{0, 0, 1, 1, 0.75, 1, 1\}$. It is unstable if we hold $v_1 = 1$ V for the SO-AND, and $v_1 = -1$ V for the SO-OR. We check this

by performing linear stability analysis, constructing the Jacobian matrix, $[J(\mathbf{x})]_{ij} = \partial F_i(\mathbf{x})/\partial x_j$, where differentiation is performed symbolically. We then determine, numerically, the eigenvalues of the Jacobian for the given critical point. We then perturb the voltage on v_2 by, say, $\delta v_2 = 0.01$ V, causing the system to evolve via numerical integration to obtain the full dynamics shown in Fig. 2.2. Results from various values of the perturbation δv_2 are also reported in Fig. 2.2, explicitly showing the topological character of the solution search: the trajectories may depend strongly on perturbations, but not the critical points, hence the final solution.

The linearized equations around the critical points can be written as, $\dot{\mathbf{x}} \approx \mathbf{J}(\mathbf{x}_{cr})(\mathbf{x} - \mathbf{x}_{cr})$, which result in the trajectories $\mathbf{x}(t) \approx \mathbf{x}_{cr} + \sum_i \mathbf{v}_i e^{\lambda_i t}$. The sum is over eigenvalues λ_i and associated eigenvectors $\mathbf{v}_i$. The eigenvectors corresponding to Re $\lambda_i < 0$ and Re $\lambda_i > 0$ define the vector spaces tangent to the stable and unstable manifolds, respectively, at each critical point.

All eigenvectors with Re $\lambda_i = 0$ are associated to *center manifolds*. In our case these center manifolds arise from the indeterminacy of the internal state variables around a critical point. To illustrate this point better, consider the example shown in Fig. 2.2, where we see that the system evolves between a critical point with a spectrum $\{\text{sign}(\lambda_i)\} = \{-,-,+,0,0,0,0\}$ to a final critical point $\{-,-,0,0,0,0,0\}$, with all stable and center directions. The overall reduction of unstable directions is a general feature of the instantons. The resulting center manifolds do not change the stability profile of the critical point, but rather can be seen as the result of additional freedom the system has in order to satisfy the equilibrium condition $\mathbf{F}(\mathbf{x}) = 0$. This freedom manifests itself in the morphing of the unstable direction of the initial point to a center manifold of the final equilibrium point.

To better clarify how the dynamics of SOLGs result in the emergence of instantons, Fig. 2.3 shows the time evolution of the memristor internal state variable x_4 (see also Fig. 2.1). This internal state does not evolve until v_3 exceeds the interval values $[-1\text{V}, 1\text{V}]$, allowing current to flow in the opposite direction through that memristive element. Only then can the memristor between terminals 1 and 3 of Fig. 2.1 change its state, thus allowing a rapid variation of v_3 towards

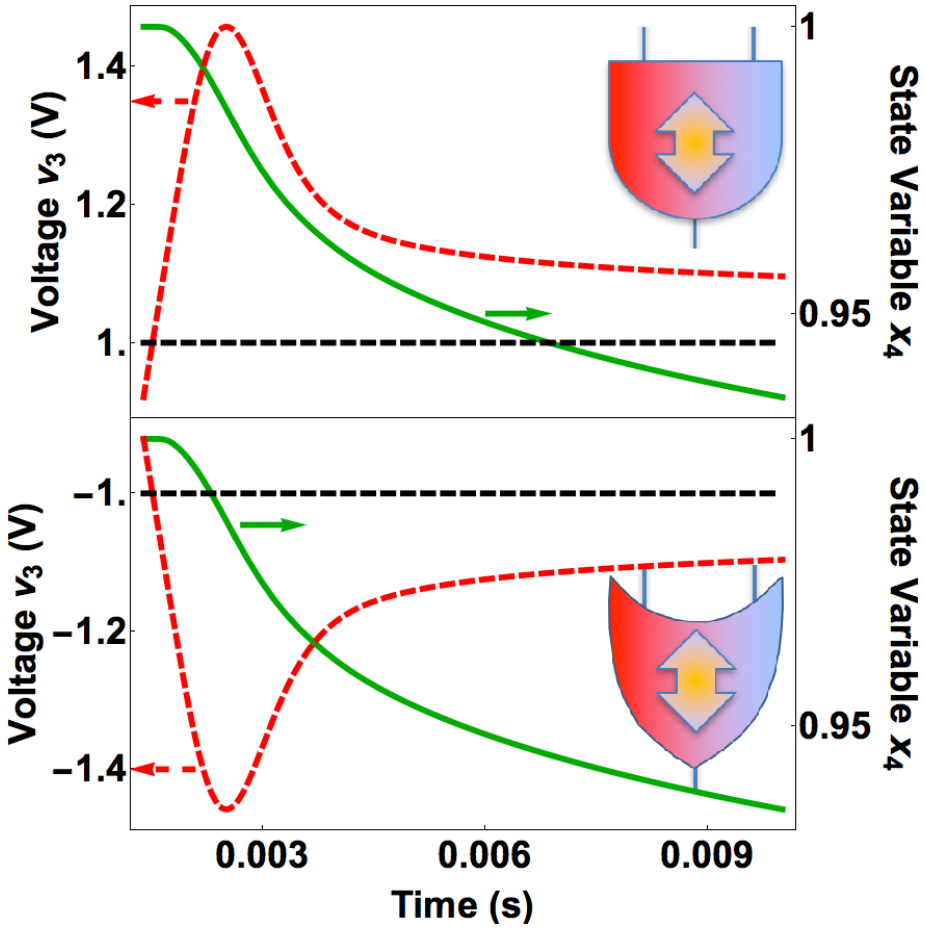

Figure 2.3: The elementary instanton in a single SOLG can best be understood from the restricted interval of the memristor state variable, $[0,1]$. The voltage shown is v_3, for a perturbation $\delta v_2 = 10^{-2}$ (see Fig. 2.2). We see that for our particular initial conditions, the voltage must leave the $[-1\text{V}, 1\text{V}]$ interval for the memristor (x_4 is shown) to change its state.

the equilibrium solution, hence the emergence of an instanton.

Finally, to show the effect of memory on the dynamics of the SOLGs, we consider the ratio R_{on}/R_{off} as a measure of memory. For $R_{on}/R_{off} \to 1$, the system has vanishing memory; for $R_{on}/R_{off} \to 0$, the system approaches infinite memory. In Fig. 2.4, we see, as R_{on}/R_{off} approaches either of the limiting values, the dynamics of the SOLGs slow down in reaching their logically-consistent (equilibrium) solution. (Equilibrium time is defined by the terminal voltages being within 1% of steady-state values after the initial dynamics have settled.) In the limit of vanishing memory, the system lacks a mechanism to inject current, therefore, the SOLG loses

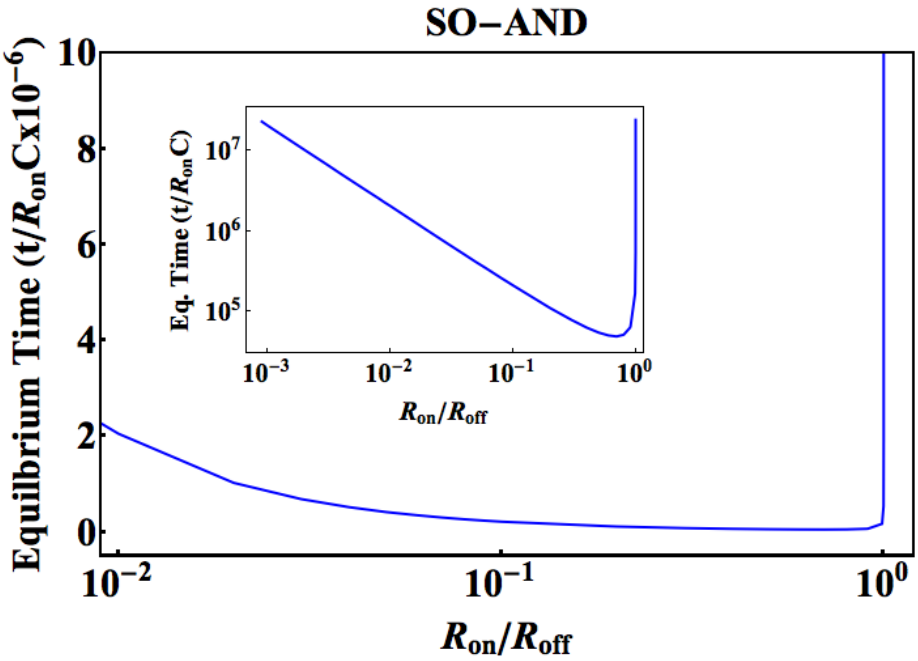

Figure 2.4: The memory content of the SOLGs (R_{on}/R_{off}) affects the time interval necessary to achieve equilibrium. The dependence of equilibrium time on memory content is illustrated in the main panel on a linear-log scale. Equilibrium is defined as the time necessary for voltages to be within 1% of their steady-state values for a given ratio R_{on}/R_{off} ($R_{on} = 0.01\ \Omega$ held fixed). The inset further illustrates the dependence on a log-log scale. The memory vanishes as $R_{on}/R_{off} \to 1$, causing dynamics to slow to a halt, while the system approaches infinite memory as $R_{on}/R_{off} \to 0$, causing the dynamics to slow as well.

the ability to self-organize. In the opposite limit of infinite memory, the dynamics are slowed due to the system possessing too many pathways to explore. The inset of Fig. 2.4 shows there is an optimum ratio for speeding up the dynamics, which will depend on the particular physical systems used to implement these gates.

Note that we have thus far assumed the system has found itself in an unstable critical point. An important question is how the SOLGs find their way to an unstable critical point from an arbitrary initial condition at $t = 0$. The very presence of one or more unstable directions would make those critical points repulsive to the system, *unless* the real part of the unstable eigenvalues were much smaller than the real part of the stable eigenvalues. This is indeed what we find in our simulations. For instance, for the critical point described above, the largest stable eigenvalue

is of the order of $\sim 10^2$, and the magnitude of the unstable eigenvalue is $\sim 10^{-3}$. We find even larger orders of magnitude differences for the other critical points we have analyzed. This makes these critical points *almost* attractive, or at least not repulsive enough to prevent the system from falling into them.

2.4 SOLGs with noise

As further evidence of topological robustness, we test the performance of the SOLGs under the influence of additive noise, modeling the internal noise of the memristors. The compact representation of the system is reformulated as,

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)) + \xi(t), \tag{2.7}$$

where $\xi = \{0, 0, \xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$. That is, the additive noise appears only in the equations for the memristor state variables,

$$\frac{d}{dt}x_j = -\alpha h(x_j, v_{M_j}) g(x_j) v_{M_j} + \xi_j(t), \tag{2.8}$$

where $\xi_j(t)$ is a white noise process of intensity Γ, with properties,

$$\langle \xi_j(t) \rangle = 0, \quad \langle \xi_i(t) \xi_j(t') \rangle = \Gamma \delta(t - t') \delta_{i,j}. \tag{2.9}$$

Simulations of the SO-AND gate with noise are shown in Fig. 2.5. Each curve is the average of 100 simulations, with each simulation having the same parameters and initial conditions as curve (*i*) in the top-left panel of Fig. 2.2. Notice that the error bars are largest after the voltage peak in the dynamics, then the error overall decreases as time increases. This is consistent with the larger instability of the trajectory when it changes more rapidly.

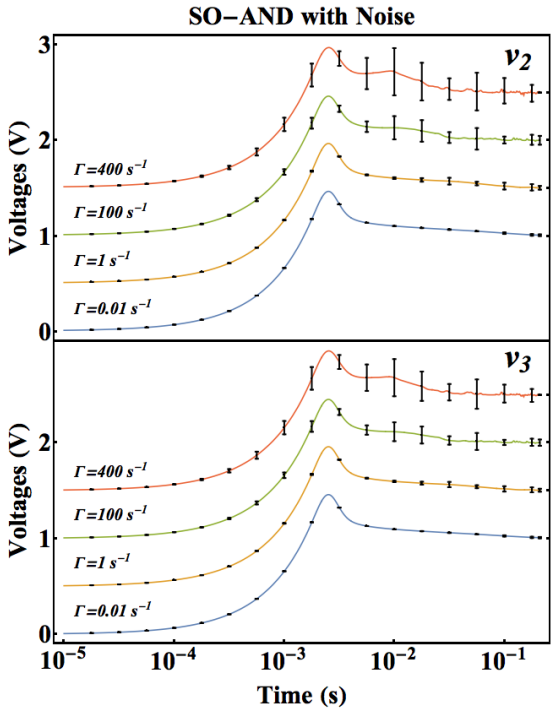

Figure 2.5: SOLGs are topologically robust in the presence of noise, meaning that the critical points connected by the instanton are unchanged, though the trajectory is modified. Using the same initial conditions as used in curve (*i*) in the top-left panel of Fig. 2.2, we have added white noise of varying strength to the simulation of the SO-AND gate's memristors. The terminal voltages are shown: v_2 (top panel) and v_3 (bottom panel). Each curve is the average of 100 simulations, with curves being translated upward for the purpose of clarity. Error bars have a height of 2 standard deviations, with some so small that they appear to be horizontal black lines. The ticks on the right axes mark the location of 1 V with respect to each corresponding curve. Note that dynamics are shown up to the equilibrium time associated with $R_{on}/R_{off} = 0.01$ in Fig. 2.4.

Increasing the noise intensity beyond the $\Gamma = 400$ s^{-1} value results in dynamics that are no longer associated with the phase space specified above. This can be understood by recalling that the function $h(x, v_M)$ in Eq. (2.1) cuts off the memristor dynamics to enforce $x \in [0, 1]$. By increasing the noise level beyond $\Gamma = 400$ s^{-1} (with $R_{on}/R_{off} = 10^{-2}$), the internal states are driven far beyond the physical limit of $x \in [0, 1]$. For example, in a TiO_2 memristor [SSSW08], the state variable is a measure of oxygen vacancies in the semiconductor film, so the boundaries of the state variable are well-defined. A noise of such an intensity as to move the state variables beyond their bounds, would imply a physical destruction of the device. However, it is clear

from Fig. 2.5 that even at a high level of intensity, the noise has not destroyed the critical points, confirming that in order to change the number and character of the critical points, the topology has to change drastically.

To make contact with actual experiments we estimate the temperature as a function of the noise strength. We, again, refer to TiO_2 memristors [SSSW08]. In that case, the intensity of the noise is related to the diffusion coefficient,

$$D = \frac{\Gamma L^2}{2}, \tag{2.10}$$

where L is the length of the oxide region of the memristor. Additionally, the diffusion coefficient is related to temperature T,

$$D = D_0 exp\left(-\frac{E_v}{k_B T}\right), \tag{2.11}$$

where $D_0 = 10^{-3}$ cm^2/s is the maximal diffusion coefficient, $E_v = 0.5$ eV is the activation energy for oxygen vacancy diffusion, and we take $L = 100$ nm [SD12, RSR99]. Therefore, we can associate a temperature with the noise strength Γ. We find $T = 271$ K for $\Gamma = 0.01$ s^{-1}; $T = 345$ K for $\Gamma = 1$ s^{-1}; $T = 475$ K for $\Gamma = 100$ s^{-1}; $T = 536$ K for $\Gamma = 400$ s^{-1}. These temperature estimates indicate that the largest value of the noise intensity is likely within the parameters of physical instability of these types of memristors.

2.5 Conclusions

In this work we have shown that the recently suggested self-organizing logic gates (which, unlike standard uni-directional Boolean gates, are "terminal-agnostic") use instantons to slice through the (large) phase space to find the stable equilibria corresponding to the consistent logical solutions of the Boolean gate they represent. The elementary instantons that are generated during the dynamics of these gates directly connect unstable initial-state critical points with the stable

equilibrium points, and eliminate the unstable directions by morphing them into center manifolds. The stable equilibria are the result of the parameter freedom of the internal state variables.

We have also demonstrated explicitly that the memory content of these gates only changes the time scale to reach the logically-consistent equilibria, while perturbations and noise can only change the trajectories in phase space but not the initial and final critical points. This implies again the topological robustness of these SOLGs.

This work then provides a better understanding of self-organizing logic, and may prove useful in the design of other practical realizations of this framework. Additionally, this work could explain other types of bi-directional logic that are being developed in the context of unconventional computing.

2.6 Remarks

In Ch. 4 and Ch. 5, implementations of self-organizing OR gates will be used to build a SOLC for the purpose of solving a particular form of Boolean satisfiability problems termed 3-SAT. In the next chapter, the Boolean satisfiability problem is defined, followed by brief descriptions of successful algorithms applied to Boolean satisfiability problems.

Chapter 2, in full, is a reprint of the material as it appears in Instantons in Self-Organizing Logic Gates in Physical Review Applied, 2018, Bearden, Sean R.B.; Manukian, Haik; Traversa, Fabio L.; Di Ventra, Massimiliano, American Physical Society, 2018. The dissertation author was the primary researcher and author of this publication.

Chapter 3

Boolean Satisfiability Problems

Now, with the self-organizing logic gates formulated with the concepts of Ch. 2, we can build a digital memcomputing machine for solving constraint satisfaction problems, specifically, Boolean satisfiability problems [Pet15]. In this chapter, the formulation of the SAT problem will be covered, along with its importance in computational complexity theory.

3.1 The SAT Problem

The Boolean satisfiability problem [Pet15] (SAT) is an important decision problem solved by determining if a solution exists to a Boolean formula. An instance of the SAT problem is satisfiable when there exists an assignment of Boolean variables (each either TRUE or FALSE) that results in the Boolean formula returning TRUE. Apart from its academic interest, the solution of SAT instances is required in a wide range of practical applications, including, travel, logistics, software/hardware design, etc. [GJ90, MS08].

The SAT problem has been studied for decades, and has an important role in the history of computational complexity theory. Computer scientists, while categorizing the efficiency of algorithms, defined the NP class for difficult decision problems [Coo71, GJ90]. Some are known as *intractable* problems, meaning they are "hard" in the sense that all known algorithms cannot be

bounded in polynomial time when determining if a solution exists in the worst-case scenario. The SAT problem was the first to be shown to belong to the class of NP-complete problems [Coo71], implying that any decision problem in NP is reducible to a SAT problem in polynomial time. There are no known polynomial-time algorithms for solving an NP-complete problem, though there are exponential time algorithms that are efficient for special cases of problem structure [GJ90]. There is a "widespread belief" [GJ90] that creation of a polynomial-time algorithm is impossible, but this belief does not limit the realization of a polynomially-bounded, *continuous-time* physical system, e.g., a digital memcomputing machine.

NP-completeness is not exclusive to SAT, with hundreds of other NP-complete problems ranging from those of academic interest (graph theory, algebra and number theory, mathematical programming) to industry application (network design, data storage and retrieval, program optimization) [GJ90]. If a polynomial-time algorithm can solve any NP-complete problem class, then all NP problems can be computed efficiently.

In this dissertation, the focus will be on the 3-SAT problem due to its popularity. The 3-SAT problem is NP-complete and a special case of SAT [GJ90]. Randomly-generated 3-SAT instances are known to be difficult to many solution methods because they lack an exploitable problem structure. For instance, one lauded algorithm, survey inspired decimation (SID), performs well on large instances of uniform random 3-SAT in the "hard regime" [MPZ02], but performs poorly in what is known as the "easy regime" [Par03]. Additionally, the focus will be on the 3-SAT problem due to it being a subclass of SAT with a consistent formulaic representation (three literals per clause).

3.2 Boolean Formula of SAT

An instance of SAT is a Boolean formula with three components [GPFW99]:

- A set of *N Boolean variables*: y_1, y_2,...,y_N.

- A set of *literals*. A literal is a variable ($l = y$) or a negation of a variable ($l = \bar{y}$).
- A set of M distinct *clauses*: C_1, C_2,..., C_M. Each clause consists of literals combined by logical OR connectives.

SAT is the decision problem of determining if an assignment of variables exists for which the Boolean formula returns TRUE, that is, all clauses must evaluate to TRUE as they are connected by AND operators to create the Boolean formula. If a such a solution exists, we say the SAT instance (Boolean formula) is satisfiable, otherwise, the instances is unsatisfiable. Commonly, it is said the instance is SAT [1]or UNSAT, respectively.

3.2.1 3-SAT

A 3-SAT instance is formulated in *conjunctive normal form* (CNF) when each clause, C_m, has three distinct literals[2] for all $m \in [1,2,...,M]$. The 3-SAT instance is satisfiable when a variable assignment can be found such that every clause has at least one literal that evaluates as TRUE.

3.2.2 Uniform Random 3-SAT

A uniform random 3-SAT instance is generated by forming M unique clauses from a set of N variables, where no variable is repeated within a clause and variables are selected with uniform probability[3]. A relevant control parameter for generating 3-SAT instances is the ratio of clauses to variables, $\alpha_r = M/N$. This ratio indicates the *constraintedness* of the formula in CNF. There is a well-known, first-order phase transition in the thermodynamic limit ($N \to \infty$): as

[1]The abbreviation SAT has different usage depending on the context in which it is used. When SAT is used in the context of a satisfiable instance, it will be paired with UNSAT for clarity. For example, the SAT/UNSAT transition [GW94].

[2]Repeated appearance of a variable within a clause is not allowed. However, it is sometimes possible-though-improbable in some generators of 3-SAT instances.

[3]When complete, one may find the instance has less than the N variables, and the value of N must be updated so the constraintedness, α_r, remains a relevant indicator of difficulty.

the random 3-SAT constraintedness is increased the probability that an instance has a solution approaches zero. The SAT/UNSAT phase transition occurs near $\alpha_r \simeq 4.267$ as $N \to \infty$, above which randomly generated instances will almost certainly be unsatisfiable, and below which instances are almost certainly satisfiable [HR04]. In Sec. 3.3.2, we will see this property can be exploited when generating benchmark instances.

3.3 Benchmarks

An obvious issue with benchmarking SAT solvers is creating instances with known solutions. Proving a 3-SAT instance is satisfiable is easy if one has the solution. However, proving that an instance is unsatisfiable generally requires more computational effort. Below are some methods of creating satisfiable benchmark instances.

3.3.1 Planted-Solutions Instances

Planting a solution so it is known to exist is a common practice [BHL$^+$02, JMS07, AJM05, KZ09, KMZ12]. However, the practice of planting solutions to generate satisfiable 3-SAT formulae does not necessarily represent the difficulty of the subset of satisfiable instances from the 3-SAT problem[4]. There are many methods found scattered among the literature, each with varying degrees of difficulty in the instances generated.

Planting a solution does not mean mean the resulting CNF will be difficult to solve. Shown in the analysis of Ref. [BHL$^+$02], there can exist local fields that guide local-search solvers[5]. Even when the average local field is eliminated, planted-solution instances may not be difficult to solve [BHL$^+$02].

Ideally, we want the instances to jump from paramagnetic phase to ferromagnetic phase at

[4]See model $1/7$ in Ref. [BHL$^+$02] or the naive generator in [HR04].

[5]Local-search solvers make repeated attempts to reduce the number of violated constraints, combined with procedures for escaping local minima.

the 3-SAT complexity peak ($\alpha_r \simeq 4.27$) with the discontinuous appearance of a *backbone* [HR04]. The backbone is the faction of the N variables in the instance that have a consistent assignment (TRUE or FALSE) for all solutions to the instance. Understood simply, if one or more of the backbone variables are not correctly assigned, it is impossible to find a solution until the backbone is corrected.

3.3.2 Probably Satisfiable Instances

If one generates large 3-SAT instances with constraintedness to the left of the complexity peak ($\alpha_r < 4.27$), there is a high probability that the instance is satisfiable, implied from the SAT/UNSAT phase transition in the thermodynamic limit. This method is used to generate large benchmarks in Ref. [BMZ05]. While the method has a finite probability of generating an unsatisfiable instance, if one solves all generated instances, as in Ref. [BMZ05], then the uncertainty no longer remains.

3.3.3 Filtered Instances

A perilous method for generating satisfiable 3-SAT instances is to generate them randomly, and filter out satisfiable instances by using a known-to-be-successful SAT solver (MiniSAT [ES03] in Ref. [MKHT20]). This has the obvious drawback that the hardest satisfiable instances will likely not appear in the benchmark set.

3.4 SAT Solvers

A brief discussion of SAT solvers used or mentioned in this dissertation.

3.4.1 WalkSAT

WalkSAT is a simple attempt to create a SAT solver that can escape the local minima that pervade difficult SAT instances [SK93]. These minima are characterized by a small number of violated constraints (clauses that do not have a literal evaluating to TRUE). Attempts to satisfy those constraints via a small number of variable flips result in a greater amount of violated constraints than the previous assignment of variables.

WalkSAT is a simple, yet effective, local-search SAT solver. With some probability p, the solver will greedily assign a variable to reduce the number of violated clauses. If $p = 1$, the greedy step can fall into inescapable local minima. To avoid this issue, set $p < 1$ and define a random step that will randomly flip the assignment of a variable, regardless if more constraints become violated.

3.4.2 Survey Inspired Decimation

Perhaps the most lauded of modern algorithms for 3-SAT, survey inspired decimation (SID) has shown impressive results on very large instances. In Ref. [BMZ05], the authors claim approximately flat scalability by attempting to solve 50 random SAT instances for each pair of $N \in \{2.5\times10^4, 5\times10^4, 10^5\}$ and $\alpha_r \in \{4.21, 4.22, 4.23, 4, 24\}$. Without solving all of the tested instances, the authors claim [BMZ05], "...the convergence time of the SP algorithm basically does not grow with N (a growth like log N , which could be expected from the geometrical properties of the factor graph, is not excluded)." Survey propagation (SP) is combined with a decimation procedure to create SID. Once decimation is complete, WalkSAT, or some other algorithm, solves the reduced file. Even with the reported success, the SID algorithm is known to perform poorly on satisfiable instances with $\alpha_r \gtrsim 4.25$ [Par03], where loops in factor graphs become more frequent.

3.4.3 Competition Solvers

Solvers from SAT competitions from satcompetition.org.

MiniSAT

In the 2005 SAT competition, a version of MiniSAT established itself as a publicly-available, state-of-the-art solver [ES06]. After the 2005 competition, many entries in later SAT competitions were iterations of MiniSAT with heuristics.

YalSAT

The Random Track winner of the 2017 SAT Competition [Bie17], YalSAT, is (yet another) local search SAT solver.[6]

3.4.4 Dynamical Systems

Simulations of digital memcomputing machines fall into the category of dynamical systems. DMMs are not the first dynamcial systems applied to Boolean satisfiability problems. In Ref. [ERT11], the authors used chaotic dynamics to solve random 3-SAT and locked occupation problems. The algorithm of Ref. [ERT11] has recently been modified and employed on a GPU in Ref. [MKHT20]. It is referred to as *AnalogSAT* in the literature. (In Appx. A.2.6, it is shown that the (filtered) benchmark instances used in Ref. [MKHT20] are easily solved by a DMM implemented on CPU.)

[6]The code for the Random Track winner of the 2018 SAT Competition is available online, but did not compile when downloaded from satcompetition.org.

3.5 Remarks

In Chapters 4 and 5, and Appendix A, DMMs are used to find solutions of satisfiable 3-SAT instances. Chapter 5 and Appendix A contain simulations of the some of the SAT solvers discussed in Sec. 3.4.

Chapter 4

Critical Branching Processes in Digital Memcomputing Machines

Note: In Ref. [BPD20] the terminology of the memory variables is changed from "fast" and "slow" to "short-term" and "long-term," respectively. However, in this chapter we will still refer to fast and slow memory variables so as to be consistent with Ref. [BSD19].

Memcomputing is a novel computing paradigm that employs time non-locality (memory) to solve combinatorial optimization problems. It can be realized in practice by means of non-linear dynamical systems whose point attractors represent the solutions of the original problem. It has been previously shown that during the solution search digital memcomputing machines go through a transient phase of avalanches (instantons) that promote dynamical long-range order. By employing mean-field arguments we predict that the distribution of the avalanche sizes follows a Borel distribution typical of critical branching processes with exponent $\tau = 3/2$. We corroborate this analysis by solving various random 3-SAT instances of the Boolean satisfiability problem. The numerical results indicate a power-law distribution with exponent $\tau = 1.51 \pm 0.02$, in very good agreement with the mean-field analysis. This indicates that memcomputing machines self-tune to a critical state in which avalanches are characterized by a branching process, and that

this state persists across the majority of their evolution.

4.1 Introduction

Unconventional computing paradigms that employ physical properties to compute specific problems are emerging as an important research direction in Physics [NC10, KPP12, NVS16]. One such paradigm is *memcomputing* [DP13b, DT18], that employs time non-locality (memory) to both process and store information on the same physical location. The digital version of this paradigm has been introduced to specifically tackle combinatorial optimization problems [TD17]. Digital memcomputing machines (DMMs) can be physically realized as non-linear dynamical systems whose point attractors represent the solutions of the problem to be solved.

Since DMMs are non-quantum systems, their equations of motion can be efficiently integrated numerically. Results from these simulations have already demonstrated that DMMs perform orders of magnitude faster than traditional algorithmic approaches on a wide variety of combinatorial optimization problems [TCSD18, DT18, MTD19, TD18, STD19].

Subsequently, by employing topological field theory [DTO17], it was shown that the physical reason behind this efficiency rests on the dynamical long-range order that develops during the transient dynamics where avalanches (instantons in the field theory language) of different sizes are generated until the system reaches an attractor [STD19]. The transient phase of the solution search of DMMs therefore resembles that of several phenomena in Nature, such as earthquakes [BT89], solar flares [LH91], quenches [Pru12], etc. Since all these phenomena show scale-free properties in the probability distribution of the avalanche sizes, it is natural to ask whether DMMs would also share this property.

In this paper, we indeed show that the transient dynamics of a DMM are characterized by a *critical branching process*. We first provide a general mean-field analysis to argue that the probability distribution of the avalanche sizes should be a critical Borel distribution with

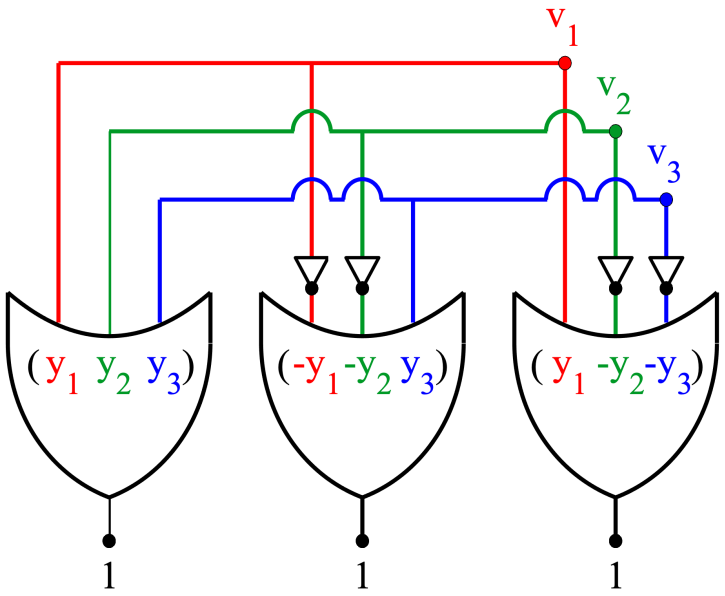

Figure 4.1: Example of a Boolean circuit, in conjunctive normal form (CNF), representing a 3-SAT. The three OR clauses (seen inside the gates) are then converted to self-organizing logic gates (SOLGs) where the propositional variables y_i are represented as electrical voltages v_i. The traditional output of the SOLG-OR is forced to be true (logical 1), because all clauses must be true to satisfy a Boolean proposition in CNF. If a literal is the negation of a variable, then the associated "input" terminal on that gate must pass through a NOT gate (triangle symbol) before the terminal is connected to other terminals sharing the same variable.

exponent $\tau = 3/2$ [ZLS95], irrespective of the problem to solve, namely it is an intrinsic feature of DMMs. We then support these results with numerical simulations of DMMs' equations of motion applied to the solution of Boolean satisfiability (SAT) instances. We have chosen to work with randomly-generated, satisfiable 3-SAT benchmark instances precisely to ensure that any feature produced by our analysis is a feature of the dynamics of DMMs, rather than a feature of the SAT instances solved.

Random 3-SAT belongs to the class of propositional logic in which a formula of Boolean variables must hold true for the problem to be satisfiable [BM10]. Propositional variables appear as literals in the formula, where a literal is a variable or its negation. Satisfiability problems are traditionally represented in "conjunctive normal form" (CNF), i.e., a conjunction (AND) of

disjunctions (OR) of literals [AB09]. A disjunction of literals is referred to as a clause. Therefore, a 3-SAT problem is one in which all clauses contain three distinct literals, of which none is a negation of the others.

A CNF formula has a simple Boolean circuit representation [AB09]. An example for a 3-SAT with three clauses is reported in Fig. 4.1. A DMM that solves the 3-SAT, say, of Fig. 4.1 can then be realized as an electrical circuit with memory (see Eqs. (4.2), (4.3), and (4.6) below) where each variable of the 3-SAT problem is represented by a voltage (we represent with $+1$ the logical 1 and with -1 the logical 0, in arbitrary units), and each traditional OR gate of Fig. 4.1 is replaced by a "self-organizing" OR gate [TD17], namely one that always attempts to dynamically satisfy the logical OR truth table at its terminals. Since the problems we are seeking to solve are satisfiable, the "out" terminals of the CNF formula in Fig. 4.1 are set to be logically true, hence the voltages at those terminals are kept fixed at +1.

4.2 Mean-field analysis

With these preliminaries we can now discuss the transient dynamics of DMMs and argue that the size of the generated avalanches (of the voltages at the gate terminals) must follow a probability distribution typical of critical branching processes. We first note that at the initial time of the dynamics a general DMM finds itself in an unstable (unsatisfied) state. The voltages at the different terminals of the gates then start evolving, and at some intervals of time some of them undergo sudden transitions, thus creating avalanches (see typical voltage trajectories in Fig. 4.2, top panel) [DTO17]. Additionally, the memory variables in DMMs have much slower dynamics than the voltage variables [TD17, DT18]. This implies that each avalanche is independent of the others generated (mean-field condition).

Now, every time a given voltage flips from +1 to -1, or vice-versa, so that its corresponding Boolean variable changes from logical 1 to logical 0, or the reverse, on average, it will only

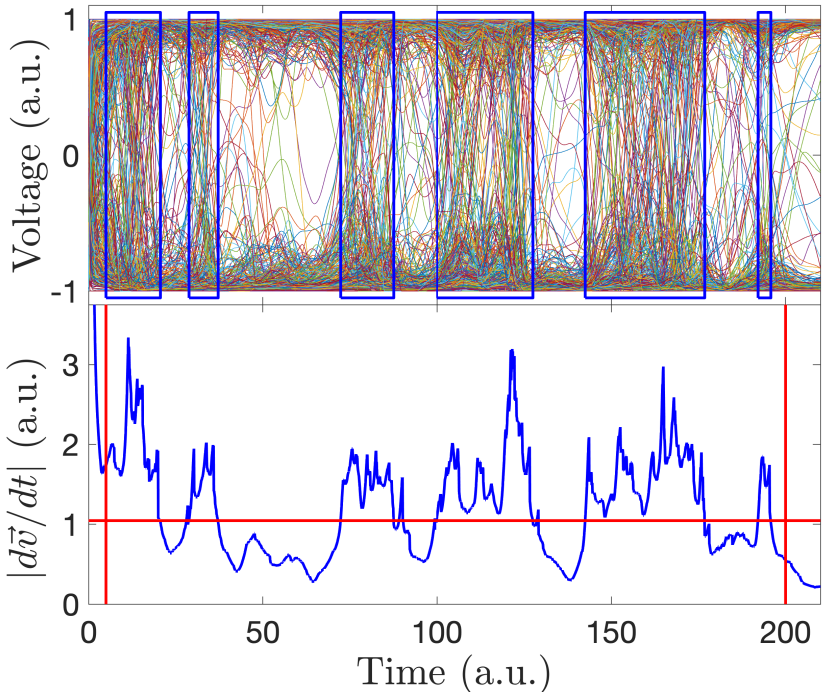

Figure 4.2: Top panel: Voltages for the solution of a 420-variable random 3-SAT problem. The blue boxed regions correspond to avalanche activity. Bottom panel: The magnitude of the vector of all voltage time derivatives. An initial short transient (finishing at the leftmost vertical red line) and the very final approach to solution (starting at the rightmost vertical red line) are ignored in the calculation of avalanches. The avalanches are identified by a threshold, determined by adding 25% of the minimum-to-maximum distance to the lowest voltage derivative value (horizontal red line). The open regions in the top panel correspond to regions of no threshold crossings of the voltage derivative.

have enough strength (power) to affect one other voltage in the circuit (its "offspring"). In turn, this "offspring" voltage, on average, will have enough strength to only affect at most one other voltage at the next time step, and so on.

Since all voltages in the circuit are equally important, the distribution of the number of voltages affected by a given voltage must be the same for each individual voltage at every time step (a "generation" step), and independent of both the number of flipped voltages at that time step and the number of affected voltages (offspring). Therefore, the flipping of a single voltage gives rise to a Poisson-distributed process where the average number of affected variables is $\mu \to 1$.

Under these conditions, the number of "descendants" of a flipped voltage (the size of the avalanches) is an integer random variable, S, described by the Borel distribution $p_S = (\mu S)^{S-1} e^{-\mu S}/S!$ [Tan61]. The expectation value of S is given by $\langle S \rangle = 1/(1-\mu)$. Therefore, due to $\mu \to 1$, DMMs must showcase a critical branching process. In fact, in the limit of $\mu \to 1$, the Stirling approximation of the Borel distribution is proportional to $S^{-3/2}$, namely a scale-free distribution [dSVBMn17, KKGvdA$^+$18].

4.3 DMM equations of motion

We can now corroborate this theoretical analysis with actual numerical results. We first design a DMM for solving 3-SAT problems. Since the choice of the dynamical system representing a DMM is not unique [TD17, DT18], we choose one very similar to the one employed in Ref. [STD19] to find the ground state of the Ising spin-glasses.

In a 3-SAT problem, clauses take the form $(\ell_i \; \ell_j \; \ell_k)$, where ℓ_i is a literal associated with a Boolean propositional variable y_i, and can be either $\ell_i = y_i$ or $\ell_i = \neg y_i$. The variables, y_i, are transformed to continuous variables, v_i, representing terminal voltages on the self-organizing OR gates (see Fig. 4.1) [TD17, BMTD18]. The voltages are bounded, $v_i \in [-1, 1]$, with $v_i \geq 0$ transformed to $y_i = 1$ and $v_i < 0$ transformed to $y_i = 0$. The negation operation used by literals is trivially performed on the voltages by multiplying them by -1.

We then convert the n-th clause to a dynamical clause by interpreting literals as voltages,

$$C_n(v_i, v_j, v_k) = \frac{1}{2}\min(1 \pm v_i, 1 \pm v_j, 1 \pm v_k), \tag{4.1}$$

where subtraction is used if $\ell_i = y_i$ and addition is used if $\ell_i = \neg y_i$, with $C_n \in [0, 1]$. When the clause is satisfied we have $C_n = 0$.

The dynamics of the voltages are influenced by the dynamical clauses in which the voltages appear [STD19],

$$\frac{d}{dt}v_i = -\sum_n x_{s,n}x_{f,n}G_{n,i}(v_i,v_j,v_k) + (1-x_{f,n})R_{n,i}(v_i,v_j,v_k), \tag{4.2}$$

where the sum is taken over all clauses, C_n, in which v_i is present. The initial condition for the voltages is chosen randomly in the interval $[-1,1]$, and the solution is found when $C_n = 0$ for all clauses. The memory variables, $x_{s,n}$ and $x_{f,n}$, along with the functions $G_{n,i}$ and $R_{n,i}$, are discussed below.

Each clause has its own memory variables, $x_{f,n}$ and $x_{s,n}$, containing information of past dynamics. The "fast" memory variable, $x_{f,n}$, determines the state of satisfiability of the clause in the recent past. Its dynamics are governed by

$$\frac{d}{dt}x_{f,n} = \beta(x_{f,n}(1-x_{f,n}))^{1/2}(C_n(v_i,v_j,v_k)-\gamma), \tag{4.3}$$

where we have chosen $\beta = 1/2$ and $\gamma = 1/8$. The fast memory variable is bounded, $x_{f,n} \in [0+\varepsilon, 1-\varepsilon]$, with the offset, $\varepsilon = 10^{-3}$, such that $x_{f,n} = 0+\varepsilon$ is interpreted to mean the clause has been satisfied for a period of time in the recent past, and $x_{f,n} = 1-\varepsilon$ means the clause has been unsatisfied for a period of time in the recent past. The offset is used to remove spurious steady-state solutions from Eq. (4.3).

The role $x_{f,n}$ has on Eq. (4.2) is to switch between the first and second terms in the summation. It can be seen that $x_{f,n}$ continuously switches between two modes: search for a satisfying assignment and hold the satisfying assignment. The first term in the summation contains a "gradient-like" function, $G_{n,i}(v_i,v_j,v_k)$, that tries to satisfy clause n by changing v_i,

$$G_{n,i}(v_i,v_j,v_k) = \frac{\partial}{\partial v_i}(1\pm v_i)\min[(1\pm v_j),(1\pm v_k)], \tag{4.4}$$

where the sign is chosen as in Eq. (4.1).

The second term in the summation of Eq. (4.2) contains a "rigidity" function [STD19], $R_{n,i}(v_i, v_j, v_k)$, which either tries to pull v_i towards an assignment ($v_i = \pm 1$) that makes $C_n = 0$ if v_i is the voltage closest to satisfying the clause, or does nothing to v_i, if v_j or v_k is the voltage closest to satisfying the clause, namely

$$R_{n,i}(v_i, v_j, v_k) = \begin{cases} v_i \pm 1, & C_n(v_i, v_j, v_k) = \frac{1}{2}(1 \pm v_i) \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

Again, the signs are chosen as in Eq. (4.1).

The slow memory variable, $x_{s,n}$, adds weight to the gradient-like term for clause n in Eq. (4.2), but does not affect the rigidity term. The additional weight promotes the ability to overcome the rigidity terms associated with other clauses. In essence, $x_{s,n}$ acts like a memory-assisted current generator that injects current into the circuit to push the DMM towards the solution, as originally conceived in Ref. [TD17]. This variable is also bounded, $x_{s,n} \in [1, M]$, where M is the number of variables in the problem[1], and we choose as its dynamics:

$$\frac{d}{dt} x_{s,n} = \alpha x_{s,n}(x_{f,n} - C_n(v_i, v_j, v_k)), \tag{4.6}$$

where we have chosen $\alpha = 1/100$. The slow memory variable will grow while its associated clause is unsatisfied, thereby giving the literals within that clause added weight to influence the dynamics of the voltages. In effect, $x_{s,n}$ contains memory of how often the clause was unsatisfied while traversing the phase space. We choose to initialize both the slow and fast memory variables as 1 for all clauses.

[1] In practice, it is not necessary to have such a large upper bound.

4.4 Results

With these DMM equations we have solved benchmark problems from previous SAT competitions. The random 3-SAT benchmarks were taken from www.satcompetition.org, and correspond to a ratio between clauses and variables of 4.3. We have solved instances of random 3-SAT for variable sizes 420, 460, 500, 540, and 600. For each variable size we have extracted data from 100 solutions found from random initial conditions. Once a solution is found, we analyze the data for avalanches.

To identify an avalanche we analyze the magnitude of the time derivative of the vector of all the voltages, $|d\vec{v}(t)/dt|$, which we refer to as the "voltage derivative". The random initial conditions cause a large spike in $|d\vec{v}(t)/dt|$, as seen in the bottom panel of Fig. 4.2. This short transient is then ignored by removing the first 5 units of time from the voltage derivative (leftmost red vertical line). When the DMM has found a solution, the voltage derivative will approach zero. We then choose to ignore also the last 10 units of time of the voltage derivative (rightmost red vertical line), so the minimum of the voltage derivative is not found at the time boundary of the simulation[2].

We show in the top panel of Fig. 4.2 all voltages in the solution of a random 3-SAT problem. From the figure it is easy to identify "open" regions that are best described by a lack of collective flipping of voltages. These regions are characteristic of an absence of avalanche events. The identified avalanche events are enclosed in blue boxes.

Once the short initial transient and the very final approach to solution are eliminated from the voltage derivative, we find its maximum and minimum. A threshold is calculated by adding 25% of the distance between minimum and maximum to the minimum of the voltage derivative. When the voltage derivative rises above the threshold we assume that an avalanche begins, and when the voltage derivative drops below the threshold the avalanche ends. Between these two

[2]Occasionally, the interval of 10 units of time is not large enough. When this occurs, the cutoff is incremented by 1 unit of time, and the minimum is checked again. The process is repeated until the minimum is not found on the time boundary.

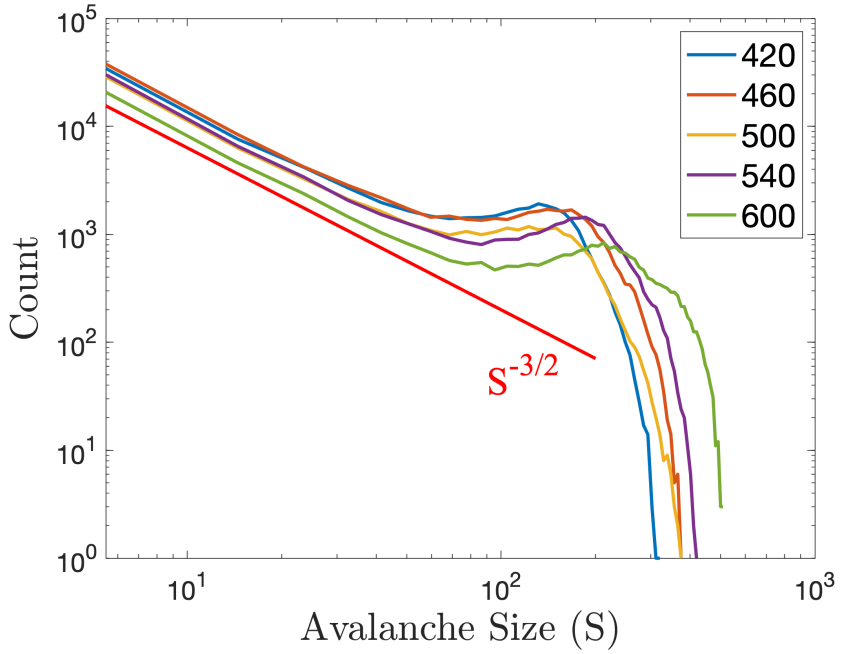

Figure 4.3: Distribution of avalanche sizes, S, for different variable sizes of random 3-SAT instances. The red line is proportional to $S^{-3/2}$ (a critical Borel distribution), and is shown for comparison. The curve for each variable size originates from 100 solutions found from Eqs. (4.2), (4.3), and (4.6), with Eq. (4.2) having random initial conditions.

events, we check how many voltages change sign. The size of the avalanche, S, is defined as the number of voltages that change sign between threshold crossings.

Our scheme for determining avalanches is, of course, not without uncertainty. For instance, it is possible for multiple avalanches to be identified as a single avalanche when they are close together in time, and thresholding may also miss the voltages that flipped immediately before and after an avalanche event.

To account for the uncertainty introduced by thresholding we choose to bin the data of our numerical simulations. The bin size, R, is chosen using the Freedman-Diaconis rule [FD81], $R = 2IQR(S)A^{-1/3}$, rounded to the nearest integer, where $IQR(S)$ is the interquartile range of the distribution of avalanche sizes, S, and A is the number of avalanches observed. For the 600-variable instances, the Freedman-Diaconis rule rounds to $R = 9$. For better comparison, we have applied this bin size to all variable sizes.

The results of the analysis are shown in Fig. 4.3. Using a power-law fit we find that the initial portion of the distribution is proportional to $S^{-\tau}$, with $\tau = 1.51 \pm 0.02$, which is in very good agreement with the one predicted by our mean-field theory, thus giving support to the hypotheses made in that analysis.

Note that the distribution in Fig. 4.3 has the characteristic "bump" seen in many finite-size power-law distributions [Pru12]. We attribute the bump to mis-identifications made by the thresholding process. For example, in Fig. 4.2 we see for $100 < t < 125$ the thresholding has identified one avalanche. If the thresholding were to be raised slightly, the thresholding would identify three avalanches. However, we have found that changing the thresholding range has a negligible effect on the scale-free trend before the bump, because the composite avalanches are comprised of smaller avalanches, and thus different thresholds simply correspond to different sampling from the same distribution.

4.5 Conclusions

In conclusion, we have provided analytical arguments, supported by numerical results, that memcomputing machines (machines that use memory to process information) undergo a critical branching process with exponent $3/2$ during their transient dynamics. The dynamics of DMMs then share some of the same features observed in many non-equilibrium phenomena encountered in Nature, and demonstrate the rich phenomenology these dynamical systems showcase, which is behind their ability to solve complex problems efficiently.

4.6 Remarks

Note that Eq. 4.6 has approximately exponential growth/decay due to the slow (long-term) memory variable appearing on the right-hand side of the equation. The form of Eq. 4.6 is not

necessary to solve these competition problems, however, it can speed up simulations. In the next chapter, the form of the DMM equations are modified to eliminate what could be an exponential energy cost. In the supplementary information (Appx. A.2.5) of the research that comprises Ch. 5, more competition instances will be solved.

Chapter 4, in full, is a reprint of the material as it appears in Critical Branching Processes in Digital Memcomputing Machines in Europhysics Letters, 2019, Bearden, Sean R.B.; Sheldon, Forrest C.; Di Ventra, Massimiliano, IOP Publishing, 2019. The dissertation author was the primary researcher and author of this publication.

Chapter 5

Efficient Solution of Boolean Satisfiability Problems with Digital Memcomputing

Note: There is supplementary information in Appendix A that is referenced throughout this chapter. The analytical arguments used to investigate the dynamics of the DMM (Appx. A.3-A.11) are the contributions of coauthors Yan Ru Pei and Dr. Massimiliano Di Ventra.

Boolean satisfiability [Pet15] is a propositional logic problem of interest in multiple fields, e.g., physics [HR04], mathematics [Mar09], and computer science [GJ90]. Beyond a field of research, instances of the SAT problem, as it is known, require efficient solution methods in a variety of applications [MS08]. It is the decision problem of determining whether a Boolean formula has a satisfying assignment, believed to require exponentially growing time for an algorithm to solve for the worst-case instances [GJ90]. Yet, the efficient solution of many classes of Boolean formulae eludes even the most successful algorithms, not only for the worst-case scenarios, but also for typical-case instances [HHW96]. Here, we introduce a memory-assisted physical system [DT18] that, when its non-linear ordinary differential equations are integrated numerically, shows evidence for polynomially-bounded scalability while solving "hard" planted-solution instances of SAT, known to require exponential time to solve in the typical case for both

complete and incomplete algorithms [HR04]. Furthermore, we analytically demonstrate that the physical system can efficiently solve the SAT problem in *continuous* time, without the need to introduce chaos or an exponentially growing energy. The efficiency of the simulations is related to the collective dynamical properties of the original physical system that persist in the numerical integration to robustly guide the solution search even in the presence of numerical errors. We anticipate our results to broaden research directions in physics-inspired computing paradigms ranging from theory to application, from simulation to hardware implementation.

5.1 Introduction

The Boolean satisfiability problem [Pet15] (SAT) is an important decision problem solved by determining if a solution exists to a Boolean formula. A SAT instance is satisfiable when there exists an assignment of Boolean variables (each either TRUE or FALSE) that results in the Boolean formula returning TRUE. Apart from its academic interest, the solution of SAT instances is required in a wide range of practical applications, including, travel, logistics, software/hardware design, etc. [MS08].

The SAT problem has been studied for decades, and has an important role in the history of computational complexity. Computer scientists, while categorizing the efficiency of algorithms, defined the NP class for difficult decision problems [Coo71, GJ90]. Some are known as *intractable* problems, meaning they are "hard" in the sense that all known algorithms cannot be bounded in polynomial time when determining if a solution exists in the worst-case scenario. The SAT problem was the first to be shown to belong to the class of NP-complete problems [Coo71], implying that any decision problem in NP is reducible to a SAT problem in polynomial time. There are no known polynomial time algorithms for solving an NP-complete problem, though there are exponential time algorithms that are efficient for special cases of problem structure [GJ90]. There is a "widespread belief" [GJ90] that creation of a polynomial time algorithm is impossible, but

this belief does not limit the realization of a polynomial *continuous-time* physical system.

NP-completeness is not exclusive to SAT, with hundreds of other NP-complete problems ranging from those of academic interest (graph theory, algebra and number theory, mathematical programming) to industry application (network design, data storage and retrieval, program optimization) [GJ90]. If a polynomial time algorithm can solve any NP-complete problem class, then all NP problems can be computed efficiently. The 3-SAT problem is NP-complete and a special case of SAT [GJ90]. Randomly-generated 3-SAT instances are known to be difficult to many solution methods because they lack an exploitable problem structure. For instance, one lauded algorithm, survey inspired decimation (SID), performs well on large instances of uniform random 3-SAT in the "hard regime" [MPZ02], but performs poorly in what is known as the "easy regime" [Par03]. We focus on the 3-SAT problem in the following due to it being a subclass of SAT with a consistent formulaic representation (three literals per clause).

5.2 Physics-inspired approach to computing

A research direction that has been far less explored concerns the solution of SAT using non-quantum *dynamical systems* [SBHF99, ERT11, ZC92, TD17]. The idea behind this approach is that the solutions of the SAT instance are mapped into the equilibrium points of a dynamical system. If the initial conditions of the dynamics belong to the basin of attraction of the equilibrium points, then the dynamical system will have to "fall" into these points. The approach is *fundamentally* different from the standard algorithms because dynamical systems perform computation in *continuous time*. Numerical simulation of continuous-time physical systems, an algorithm, requires the discretization of time to integrate the ordinary differential equations (ODEs) representing the physical system. As such, the dynamical-systems approach is ideally suited for a hardware implementation.

The authors of Ref. [ERT11] have shown that an appropriately designed dynamical system

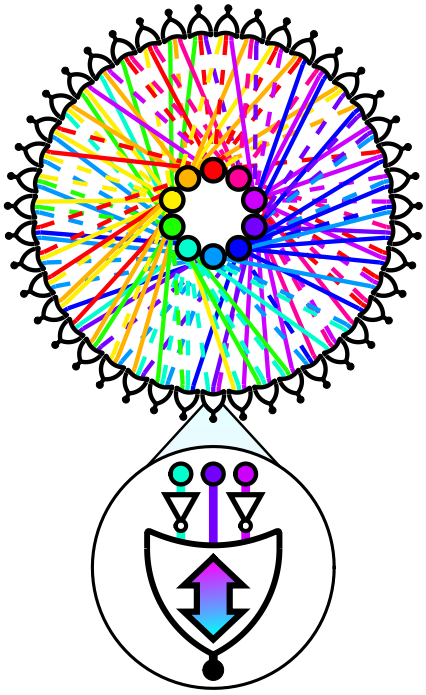

Figure 5.1: Schematic of a self-organizing logic circuit representing a 3-SAT instance. The circuit is created from the constraints of a 3-SAT formula consisting of $N = 10$ variables, and $M = 43$ clauses. The formula is converted into 10 voltage nodes (inner nodes) and 43 self-organizing OR gates [BMTD18]. The black nodes traditionally associated with the output of the OR gates are fixed to TRUE to enforce the constraints. Dashed lines in the circuit represent NOT gates on the OR gate terminals. Ignoring the black nodes, the circuit can be interpreted as a factor graph with the gates becoming function nodes (see also Fig. 5.3). The clause represented by the highlighted self-organizing OR gate is $(\bar{y}_i \vee y_j \vee \bar{y}_k)$, where NOT gates are used for logical negation. The double-headed arrow indicates this is a self-organizing logic gate with no distinction between an input and an output (terminal agnosticism). The circular representation of the linear circuit is a reminder that the ordering of gates is irrelevant to the solution search.

can find the solutions of hard 3-SAT instances in continuous polynomial time, however, at a cost of *exponential* energy fluctuations. The reason for this exponential energy cost can be traced to the transient *chaotic* dynamics of the dynamical systems proposed in Ref. [ERT11]. As the problem size grows, the chaotic behavior translates into an exponentially increasing number of integration steps required to find the equilibrium points of the corresponding ODEs.

5.3 The digital memcomputing approach

In recent years, a different physics-inspired computational paradigm has been introduced, known as *digital memcomputing* [TD17, DT18]. Digital memcomputing machines (DMMs) are non-linear dynamical systems specifically designed to solve constraint satisfaction problems, *e.g.*, 3-SAT, with the assistance of memory [TD17] (Fig. 5.1). The only equilibrium point(s) of the DMM is the solution(s) of the original problem. However, unlike previous work, DMMs are designed so that they have no other equilibrium points; see Appx. A.6.4. Additionally, the dynamics will never enter a periodic orbit or a state of chaos [DT17a] (see Appx. A.9).

The ability of continuous time dynamics to perform the solution search without resorting to chaotic dynamics results in efficient *simulations* (an algorithmic implementation) of DMMs using computationally-inexpensive integration schemes and modern computers. In addition, it was shown that DMMs find the solution of a given problem by employing topological objects, known as instantons, that connect critical points of increasing stability in the phase space [DTO17, DO19a] (see Appx. A.11). Simulations found the DMMs then self-tune into a critical (*collective*) state which persists for the whole transient dynamics until a solution is found [BSD19]. It is this critical branching behavior that allows DMMs to explore *collective* updates of variables during the solution search, without the need to check an exponentially-growing number of states. This is in contrast to local-search algorithms which are characterized by a "small" (not collective) number of variable updates at each step of the computation [HR04].

Here, we introduce a physical DMM to find solutions of the 3-SAT problem. So as to facilitate the reading of our paper, we have contained the mathematical description of our physical DMM to Box 1. We then perform numerical simulations of the ODEs (discretized time) of the DMM to solve random 3-SAT instances with planted solutions. These instances are generated with a clause distribution control (CDC) procedure, known to require exponentially growing time to solve in the typical case for both complete and incomplete algorithms [BHL$^+$02]. The

CDC instances have found use as benchmarks in recent years of SAT competitions (satcompetition.org) [Bal16, Heu17, Heu18]. The simulations have been performed using a forward-Euler integration scheme [Sau12] with an adaptive time step, implemented in MATLAB R2019b with each solution attempt run on a single logic core of an AMD EPYC 7401 server (see also Appx. A.2).

We compare our results with those obtained from two well-known algorithms: WalkSAT, a stochastic local-search procedure [SK93], and survey-inspired decimation (SID), a message-passing procedure utilizing the cavity method from statistical physics [MPZ02]. (in Appx. A.2 we also compare with the winner of a recent SAT competition and AnalogSAT [MKHT20]. Comparison is achieved via scalability of some indicator vs. the problem size. As expected, both algorithms show an exponential scaling for the CDC instances (Fig. 5.2). Our simulations instead show a power-law scalability of integration steps ($\sim N^a$) for typical cases, where the typical case is inferred from the median number of integration steps.

Finally, we show that the dynamics is capable of finding satisfying variable assignments for 3-SAT in polynomially-bounded (linear or sub-linear) *continuous* time without the need of an exponentially increasing energy cost demonstrated via certain dissipative and topological properties of the system (see Appx. A.9-A.10).

While the reported numerical and analytical results do not resolve the famous P vs. NP debate, (which, incidentally, is formulated for Turing machines, that compute in *discrete*, not continuous time) they show the tremendous advantage of physics-based approaches to computation over traditional algorithmic approaches.

5.4 DMM for 3-SAT

The 3-SAT formula is constructed by applying conjunction (AND), disjunction (OR), and negation (NOT) operations to Boolean variables (TRUE or FALSE), with parentheses used

to indicate the order of operations [MZ09]. A formula contains N Boolean variables (y_i), M clauses, and $3M$ literals. Each clause (constraint) consists of three literals connected by logical OR operations, i.e., $(l_i \vee l_j \vee l_k)$, where a literal, l_i, is simply one of the Boolean variables ($l_i = y_i$) or its negation ($l_i = \bar{y}_i$). A clause is satisfied if at least one literal is TRUE (OR operations), and the formula is satisfiable if all clauses (AND operations) are simultaneously satisfied. The complexity of the problem emerges from the interaction among constraints, and is observed in the well-studied easy-hard-easy transition in 3-SAT, where easy and hard regimes are identified by the ratio $\alpha_r = M/N$, with the complexity peak (hardest instances) occurring around $\alpha_r = 4.27$ [GW94].

To construct a DMM that finds a satisfying assignment for 3-SAT we follow the general procedure outlined in Ref. [DT18]. To begin, the Boolean variables, y_i, are transformed into continuous variables for use in the DMM. The continuous variables can be realized in practice as voltages on the terminals of a self-organizing OR gate [TD17]. Such a gate can influence its terminals to push voltages towards a configuration satisfying its OR logic *regardless* of whether the signal received by the gate originates from the traditional input or the traditional output (see Fig. 5.1). The voltages are bounded, $v_i \in [-1, 1]$, with Boolean values recovered by thresholding: TRUE if $v_i > 0$, FALSE if $v_i < 0$, and ambiguous if $v_i = 0$. To perform the logical negation operation on the continuous variable, one trivially multiplies that quantity by -1. The self-organizing logic circuit that comprises the DMM is built by connecting all of the self-organizing OR gates (see Fig. 5.1). See Appx. A.3.1 for an extended discussion of the thresholding procedure for the voltages.

Next, we interpret a Boolean clause as a dynamical constraint function, with its state of satisfaction determined by the voltages. The *m-th* Boolean clause, $(l_{i,m} \vee l_{j,m} \vee l_{k,m})$, becomes a constraint function,

$$C_m(v_i, v_j, v_k) = \frac{1}{2} \min[(1 - q_{i,m} v_i), (1 - q_{j,m} v_j), (1 - q_{k,m} v_k)], \tag{5.1}$$

where $q_{i,m} = 1$ if $l_{i,m} = y_i$, and $q_{i,m} = -1$ if $l_{i,m} = \bar{y}_i$. The function is bounded, $C_m \in [0,1]$, and a clause is necessarily satisfied when $C_m < 1/2$. The instance is solved when $C_m < 1/2$ for all clauses. By thresholding the clause function we avoid the ambiguity associated with $v_i = 0$. If some voltage is ambiguous ($v_j = 0$) and all clauses are satisfied, then any Boolean assignment to y_j will be valid in that configuration. The use of a minimum function in C_m preserves an important property of 3-SAT. A clause is a constraint, and, by itself, a clause can only constrain one variable (via its literal). (Note that the minimum operation introduces some form of discontinuity to the dynamical system, for which we develop the formalism to study in Appx. A.4 and A.5.) The values of two literals are irrelevant to the state of the clause if the third literal results in a satisfied clause.

Finally, a DMM employs memory variables to assist with the computation [TD17, DT18]. The memory variables transform equilibrium points that do not correspond to solutions of the Boolean formula into unstable points in the voltage space (see Appx. A.8), leaving the solutions of the 3-SAT problem as the only minima. We choose to introduce two memory variables per clause: short-term memory, $x_{s,m}$, and long-term memory, $x_{l,m}$. The terminology intuitively describes the behavior of their dynamics. For the short-term memory, $x_{s,m}$ lags C_m, acting as an indicator of the recent history of the clause. For the long-term memory, $x_{l,m}$ collects information so it can "remember" the most frustrated clauses, weighting their dynamics more than clauses that are "historically" easily satisfied. Both the number and type of memory variables, as well as the form of the resulting dynamical equations, are not unique provided neither chaotic dynamics nor periodic orbits are introduced [DT18].

We choose for the dynamics of voltages and memory variables the following,

$$\dot{v}_n = \sum_m x_{l,m} x_{s,m} G_{n,m} + (1 + \zeta x_{l,m})(1 - x_{s,m}) R_{n,m}, \tag{5.2}$$

$$\dot{x}_{s,m} = \beta(x_{s,m} + \varepsilon)(C_m - \gamma), \tag{5.3}$$

$$\dot{x}_{l,m} = \alpha(C_m - \delta), \tag{5.4}$$

where the summation is taken over all constraints in which the voltage appears. The memory variables are *bounded*, with $x_{s,m} \in [0,1]$ and $x_{l,m} \in [1, 10^4 M]$. The boundedness of voltage and memory variables implies that there are no diverging terms in the above equations (see Appx. A.6.2).

The parameters α and β are the rates of growth for the long-term and short-term memory variables, respectively. Each memory variable has a threshold parameter used for evaluating the state of C_m, and the two parameters are restricted to obey $\delta < \gamma < 1/2$. (This also guarantees that there is a sufficiently large basin of attraction for the solutions. See Appx. A.7 for a detailed explanation.). Eq. (5.3) has a small, strictly-positive parameter, $0 < \varepsilon \ll 1$, to remove the spurious solution ($x_{s,m} = 0$). However, ε additionally serves as a trapping rate in the sense that smaller values of ε make it more difficult for the system to flip voltages when some C_m begins to grow larger than γ.

In Eq. (5.2), the first term in the summation is a "gradient-like" term, the second term is a "rigidity" term [BSD19]. The gradient-like term attempts to influence the voltage in a clause based on the value of the other two voltages in the associated clause, $G_{i,m}(v_i, v_j, v_k) = \frac{1}{2} q_{i,m} \min[(1 - q_{j,m} v_j), (1 - q_{k,m} v_k)]$. Consider the two extremes: if the minimum results is $G_{i,m} = 1$, then v_i needs to be influenced to satisfy the clause. Conversely, if the minimum gives $G_{i,m} = 0$, then v_i does not need to influence the clause state (see Appx. A.2.1).

For the rigidity term, we choose $R_{i,m}(v_i, v_j, v_k) = \frac{1}{2}(q_{i,m} - v_i)$ if $C_m(v_i, v_j, v_k) = \frac{1}{2}(1 - q_{i,m} v_i)$, and $R_{i,m}(v_i, v_j, v_k) = 0$ otherwise [BSD19]. The purpose of the three rigidity terms for a

constraint is to attempt to hold one voltage at a value satisfying the associated *m-th* clause, while doing nothing to influence the evolution of the other two voltages in the constraint. Again, this aligns with the 3-SAT interpretation that a clause can only constrain one variable. The short-term memory variable acts as a switch between gradient-like dynamics and rigid dynamics. During the solution search, G_m will seek to influence three voltages until clause m has been satisfied. Then, as $x_{s,m}$ decays to zero, R_m takes over. The long-term memory variables weight the gradient-like dynamics, giving greater influence to clauses that have been more frustrated during the solution search. The rigidity is also weighted by $x_{l,m}$, but reduced by ζ.

5.5 Numerical results and discussion

It is important to realize that any simulation of a dynamical system is an algorithm because the continuous-time dynamics of the system must be discretized. Identifying our simulation as an algorithm invites a method to compare our results with those of popular algorithms, specifically, WalkSAT [SK93] and survey inspired decimation (SID) [MPZ02]. Before we compare results, we then need a general definition of a step.

We define an algorithmic step to be all the computation that occurs between checks of satisfiability. The WalkSAT algorithm flips one variable at a time then checks the satisfiability of the formula. Therefore, a WalkSAT step is a single variable flip. SID uses WalkSAT as part of its solution search, so the interpretation of steps is the same when SID uses WalkSAT. Prior to entering into WalkSAT, SID performs a message-passing procedure known as survey propagation [MPZ02]. In the SID implementation we used [GAE18] there is no check for satisfiability during the decimation procedure, so we generously identify the entire survey propagation with decimation as a single step. Our DMM algorithm checks the satisfiability of the formula after each time step of the integration. Of course, the amount of computation within a step may vary greatly based on the algorithm, but this does not affect comparison of the scalability. In fact, if an algorithm is

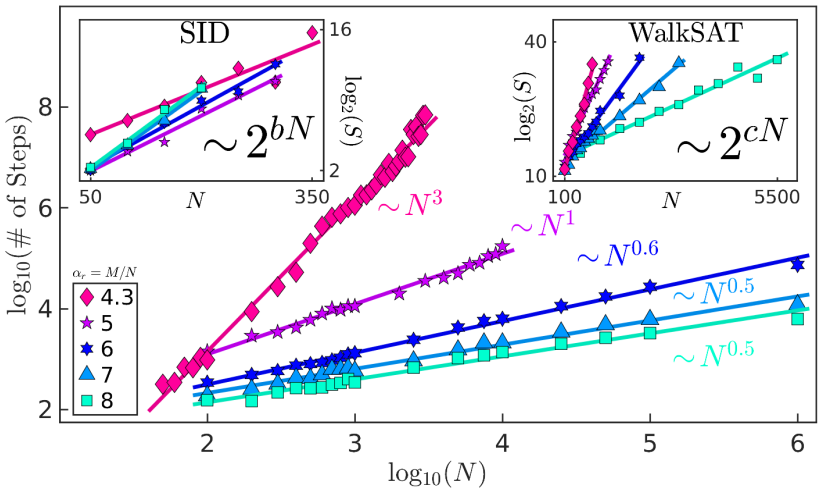

Figure 5.2: Typical case scalability of 3-SAT instances at fixed clause-to-variable ratio. In the main panel, we use our DMM algorithm to attempt to solve 100 planted-solution instances of 3-SAT per pair of α_r (clause-to-variable ratio) and N (number of variables). When we achieve more than 50 instances solved, we find power-law scalability of the median number of integration steps (typical case) as the number of variables, N, grows. (In Appx. A, we show many data points are comprised of 90 or more instances solved within the allotted steps.) The exponent values ($\sim N^a$) are $a_{4.3} = 3.0 \pm 0.1$, $a_5 = 1.00 \pm 0.05$, $a_6 = 0.63 \pm 0.03$, $a_7 = 0.48 \pm 0.03$, and $a_8 = 0.46 \pm 0.04$. The insets show exponential scalability for a stochastic local-search algorithm (WalkSAT) and a survey-inspired decimation procedure (SID) on CDC instances. (S is for number of steps.) Notice the scalability for SID has a trend opposite that seen in the DMM and WalkSAT. This is expected when one considers the increase in factor graph loops as α_r grows. For the SID scaling of $\alpha_r = 4.3$, the $N = 350$ did not achieve a median number of solutions, and is thus a lower bound. Parameters of the scaling for SID: $b_{4.3} = (3 \pm 1) \times 10^{-2}$, $b_5 = (3.7 \pm 0.7) \times 10^{-2}$, $b_6 = (4.1 \pm 0.6) \times 10^{-2}$, $b_7 = (5 \pm 1) \times 10^{-2}$, and $b_8 = (5 \pm 1) \times 10^{-2}$; for WalkSAT: $c_{4.3} = (3.2 \pm 0.3) \times 10^{-2}$, $c_5 = (1.9 \pm 0.2) \times 10^{-2}$, $c_6 = (1.2 \pm 0.1) \times 10^{-2}$, $c_7 = (7.5 \pm 0.6) \times 10^{-3}$, and $c_8 = (4.1 \pm 0.5) \times 10^{-3}$.

exponential in the number of steps, then the amount of computation within a step cannot improve its scalability. For our DMM, each step has a constant amount of computation per time step of integration. With this definition of an algorithmic step, we have a method to meaningfully compare the different algorithms.

We can now test these approaches on CDC instances with planted solutions. In Appx. A.3.3, we give an account of how these instances are generated, and why they are difficult to solve. Here, we just note that difficult CDC instances are created when $\alpha_r > 4.25$ and

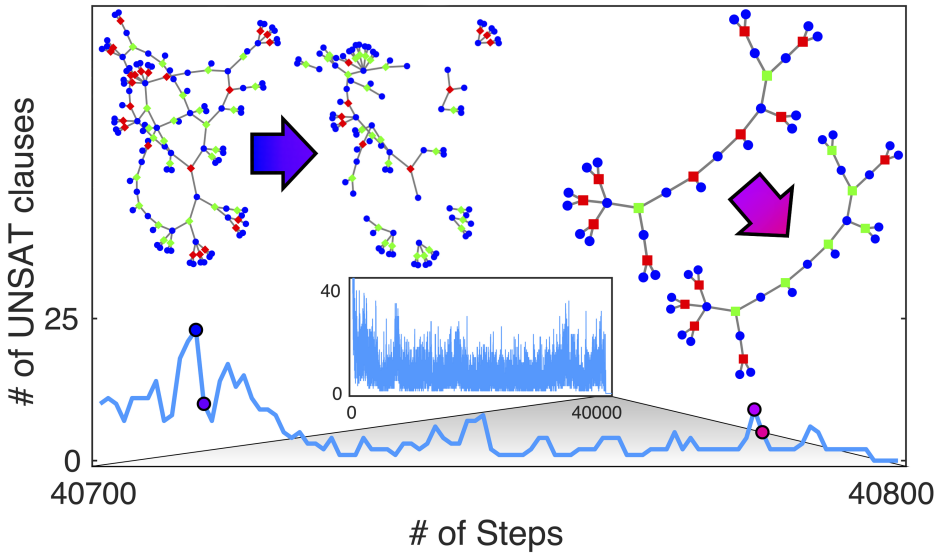

Figure 5.3: Time evolution of a typical DMM simulation showing collective updates to the solution search. The figure highlights one solution attempt of a CDC instance of size $N = 500$ at $\alpha_r = 4.3$. The inset shows the number of unsatisfied clauses during the entire solution search. The main panel zooms in on the search as the solution is approached. We choose two single integration step transitions and explore the local factor graph. The circles are the variable nodes (blue), and squares are function nodes (red if unsatisfied, green if recently unsatisfied). The transition at left is characterized by 13 clauses becoming satisfied, the transition at right results in 4 clauses becoming satisfied. Neither transition results in satisfied clauses becoming unsatisfied.

$0.077 < p_0 < 0.25$, where p_0 is the probability that the planted solution results in a clause with zero false literals [BHL$^+$02]. We have performed no preprocessing on the 3-SAT instances to reduce their size, not even the removal of pure literals (those appearing wholly negated or unnegated) [GPFW99].

We numerically integrated Eqs. (5.2), (5.3), and (5.4) with the forward-Euler method using an adaptive time step, $\Delta t \in [2^{-7}, 10^3]$. For parameters, we have used $\alpha = 5$, $\beta = 20$, $\gamma = 1/4$, $\delta = 1/20$, and $\varepsilon = 10^{-3}$. For high ratio, $\alpha_r \geq 6$, we find $\zeta = 10^{-1}$ to provide better scaling results. For ratios that approach the complexity peak, we used $\zeta = 10^{-2}$ for $\alpha_r = 5$, and $\zeta = 10^{-3}$ for $\alpha_r = 4.3$. In Fig. 5.2, we report the results for CDC instances generated with $p_0 = 0.08$. In our simulations, we expectedly find the difficulty of CDC instances increases with increasing p_0 (see Appx. A.2).

In Fig. 5.2, for the problem sizes tested, we find a power-law scaling for the median

number of integration steps for the simulations of DMMs. We also find that integration time variable (t), CPU time, and long-term memory (x_l) are bounded by a polynomial scaling, and the average step size shows power-law decay (see Appx. A.2.3). The optimized WalkSAT algorithm [Kau18] we have used instead exhibits an exponential scaling at relatively small problem sizes, confirming the previous results of Ref. [BHL$^+$02]. An exponential scaling is also observed for the SID algorithm [GAE18].

The CDC instances are structured to confuse stochastic local-search algorithms, so the exponential scaling of WalkSAT is expected (right inset Fig. 5.2). To understand the exponential performance of SID (left inset Fig. 5.2), we need to understand the success of SID on random 3-SAT. When generating uniform random 3-SAT at the complexity peak with a general method (no planted solutions), the typical case can be exploited by SID due to the existence of treelike structures in the factor graph [BMZ05]. (For those unfamiliar with factor graphs, if the factor graph was a tree, then one would be able to visually, thus easily, find the solution from the graph [MM09].) However, as demonstrated in Fig. 5.2, SID performs poorly when given a 3-SAT instance with a factor graph that is not locally treelike. It is also known that SID performs poorly at high ratios ($\alpha_r \gtrsim 4.25$) [Par03], as loops in the factor graph become more common, explaining the opposite scaling trend seen in Fig. 5.2.

To further confirm that the usefulness of our DMM algorithm on CDC instances is independent of our generation of formulae, we have solved generalized CDC instances [Bal16] used in the 2017 [Heu17] and 2018 [Heu18] SAT competitions (satcompetition.org).

Our modified competition DMM solves *all* tested competition CDC instances on its first attempt with random initial conditions, and does so within the 5000-second timeout established by the competition (see Appx. A.2.5). We find the overhead of numerical simulations of ODEs does not forbid our DMM from being competitive due to the use of the forward-Euler integration scheme.

5.6 Long-range order and analytical properties of DMMs for 3-SAT

We finally show that collective behavior (*long-range order*) [DTO17, DO19a] in DMMs is responsible for the observed efficiency in the solution search. In order to do this, it is helpful to visualize subgraphs of the factor graph generated from a 3-SAT instance. In Fig. 5.3, we visualize the change in state of local factor graphs during a single time step of integration as our DMM approaches a solution. It is apparent that the system explores many paths in the factor graph, collecting information as it does. However, unlike SID, when the DMM explores a path leading to contradiction it can *correct itself*. The factor graphs shown in Fig. 5.3 only include clauses (function nodes) that are unsatisfied (red) or recently unsatisfied (green), and all variable nodes connected to these clauses. A clause, m, is identified as recently unsatisfied if the short-term memory is $x_{s,m} > 0$ but the clause is currently satisfied. The factor graph transitions show that *collective events* occur that satisfy multiple clauses. This is in agreement with many results on DMMs for different types of problems [DTO17, STD19]. Additionally, the factor graph transition on the left of Fig. 5.3 breaks up the graph into smaller, disconnected factor graphs, making the search exponentially more efficient.

As anticipated, to strengthen these numerical results, we have also analytically demonstrated that the dynamics described by Eqs. (5.2), (5.3), and (5.4) terminate *only* when the system has found the solution to the 3-SAT problem (namely the phase space has only saddle points and the minima corresponding to the solution of the given problem; Appx. A.6 and A.7). In addition, neither periodic orbits nor chaos can coexist if solutions of the 3-SAT are present (Appx. A.9). Finally, using supersymmetric topological field theory, we have demonstrated that the continuous-time dynamics (physical implementation) reach the solution of a 3-SAT instance, for a fixed α_r, in linear or sub-linear *continuous* time, irrespective of the difficulty of the instance

(Appx. A.11).

However, note that such a scalability *does not* necessarily translate to the same scalability of the *numerical* integration of Eqs. (5.2), (5.3), and (5.4), where the discretization of time is necessary. Nevertheless, due to the absence of chaos, we empirically find that the scalability of our numerical simulations is still polynomially bounded for typical-case CDC instances.

5.7 Conclusions

We have presented an efficient dynamical-system approach to solving Boolean satisfiability problems. Along with arguments for polynomial-time scalability in *continuous* time, we have found that the *numerical* integration of the corresponding ODEs show power-law scalability for typical cases of 3-SAT instances which required exponential time to solve with successful algorithms. The efficiency derives from *collective* updates to the variables during the solution search (*long-range order*).

In contrast to previous work [ERT11], our dynamical systems do not suffer from exponential fluctuations in the energy function due to chaotic behavior. The dynamical systems we propose find the solution of a given problem without ever entering a chaotic regime, by virtue of the variables being bounded. The implication is that a hardware implementation of DMMs would only require a polynomially-growing energy cost. Our work then also serves as a counterexample to the claim of Ref. [ERT11] that chaotic behavior is necessary for the solution search of hard optimization problems. In fact, we find chaos to be an undesirable feature for a scalable approach (See Appx. A.2.6).

Although these analytical and numerical results do not settle the famous P vs. NP question, they show that appropriately designed physical systems are very useful tools for new avenues of

research in constraint satisfaction problems.

5.8 Remarks

Notice the difference between the equation of motion for the "slow memory" given in Eq. (4.6) and the long-term memory equation of motion in Eq. (5.4). In Eq. (4.6), the memory variable appears on the RHS, whereas in Eq. (5.4) it does not. As an approximation, when we assume $(C_m - \delta)$ is constant in Eq. (5.4) we see linear growth of x_l w.r.t. integration time, t. This approximate linearity removes the exponential energy cost present in Ref. [ERT11]. In the absence of exponential energy cost, the topological robustness efficiently guides numerical simulations of the solution search in the presence of noise and integration error.

In Appx. B, the DMM of this chapter is modified to perform a search of local minima in MaxSAT problems.

Chapter 5 and Appendix A, in full, have been submitted for publication. Bearden, Sean R.B.; Pei, Yan Ru.; Di Ventra, Massimiliano, 2020. The dissertation author is the primary researcher and author of the main text (Ch. 5), and Yan Ru Pei is the primary researcher and author of the supplementary information (Appx. A).

Appendix A

Supplementary Information: Efficient Solution of Boolean Satisfiability Problems with Digital Memcomputing

A.1 Summary of Major Results

For the benefit of the reader we summarize the major results presented in this Supplementary Material (SM).

- In Section A.2 we describe the numerical method and implementation we used to solve Eqs. (2)-(4) in the main text. We also show several other numerical results on additional 3-SAT instances to support the ones reported in the main text. In particular, we show that the time variable of integration, CPU time, and slow memory variables all scale as a *power law* in the size of the problem. We also show that the average time step of the integration needs only to decrease as a *power law* with increasing problem size.

- In Sections A.4 and A.5, we show that a *unique* dynamical trajectory can be constructed for the discontinuous flow field governing our dynamics. For practical purposes, the analytic

trajectory is constructed such that it is approximated by the numerical trajectory obtained with the forward Euler integration method used in our numerical analysis reported in the main text.

- In Section A.6, we show that our dynamics are *bounded* by a positive invariant compact set, and the dynamics terminate *only* when the system has found the solution to the 3-SAT problem. This guarantees a correspondence between the fixed points of the dynamics and the solutions of the 3-SAT problem, and absence of local minima.

- In Section A.7, we show that the *basin of attraction* of the solution for our flow field contains a large hypercube in the voltage space. In other words, once the trajectory has entered this region, the dynamics are guaranteed to converge to a solution.

- In Section A.9, we show the *absence of periodic orbits* in the voltage dynamics. This result, augmented by the fact that the memcomputing flow is *not topologically transitive*, implies *absence of chaos* (à la Devaney).

- In Section A.10, we show that our system is *dissipative*, in the sense that the volume of any initial set in the phase space contracts under the flow field.

- In Section A.11 we show using topological field theory that the *continuous-time* dynamics reach the fixed points in a time that scales with problem size, n, as $O(n^{\alpha})$ with $\alpha \leq 1$. This result does not necessarily apply to the numerical solution of the dynamical equations due to integration overhead and numerical noise.

A.2 Numerical implementation and additional simulation results

A.2.1 Numerics

For ease of discussion, the equations of motion for the digital memcomputing machine (DMM) are reproduced here. The *m-th* Boolean clause, $(l_{i,m} \vee l_{j,m} \vee l_{k,m})$, becomes a clause function,

$$C_m(v_i, v_j, v_k) = \frac{1}{2}\min[(1 - q_{i,m}v_i), (1 - q_{j,m}v_j), (1 - q_{k,m}v_k)], \tag{A.1}$$

where $q_{n,m} = 1$ if $l_{n,m} = y_n$, and $q_{n,m} = -1$ if $l_{n,m} = \bar{y}_n$. The DMM's equations then read:

$$\dot{v}_n = \sum_m x_{l,m}x_{s,m}G_{n,m} + (1 + \zeta x_{l,m})(1 - x_{s,m})R_{n,m}, \tag{A.2}$$

$$\dot{x}_{s,m} = \beta(x_{s,m} + \varepsilon)(C_m - \gamma), \tag{A.3}$$

$$\dot{x}_{l,m} = \alpha(C_m - \delta), \tag{A.4}$$

$$G_{n,m}(v_n, v_j, v_k) = \frac{1}{2}q_{n,m}\min[(1 - q_{j,m}v_j), (1 - q_{k,m}v_k)], \tag{A.5}$$

$$R_{n,m}(v_n, v_j, v_k) = \begin{cases} \frac{1}{2}(q_{n,m} - v_n), & C_m(v_n, v_j, v_k) = \frac{1}{2}(1 - q_{n,m}v_n), \\ 0, & \text{otherwise}, \end{cases} \tag{A.6}$$

where $G_{n,m}$ and $R_{n,m}$ equal 0 when variable n does not appear in clause m.

In Eq. (A.2), each of the N voltages (variables) are guided by M constraints (clauses). Each constraint influences three voltages simultaneously, while switching between two dynamical terms containing a gradient-like function, $G_{n,m}$, and a "rigidity" function, $R_{n,m}$.

In addition to the voltages, memcomputing utilizes memory variables to assist with the

computation. The short-term memory, $x_{s,m}$, controls the switching between $G_{n,m}$ and $R_{n,m}$. The long-term memory, $x_{l,m}$ collects information so it can "remember" the most frustrated constraints (unsatisfied clauses), weighting their dynamics more than clauses that are "historically" easily satisfied.

To understand the gradient-like function better, consider the two extremes: if $G_{n,m} = 1$, then v_n needs to be influenced to satisfy the clause. (Recall, there are three voltages associated with the *m-th* constraint, but, independent of information from other constraints, no determination can be made on which voltage needs to be influenced.) Conversely, if $G_{n,m} = 0$, then v_n does not currently need to influence the *m-th* constraint state. The purpose of the rigidity term, $R_{n,m}$, is to attempt to hold one voltage at a value satisfying the associated *m-th* constraint, but to do nothing to influence the evolution of the other two voltages in the constraint.

The long-term memory variable weights the gradient-like dynamics, giving greater influence to constraints that have been more frustrated during the solution search. The rigidity term is also weighted by $x_{l,m}$, but reduced by ζ. The parameter ζ can be thought of as a "learning rate". More difficult instances, as characterized by their clause-to-variable ratio, require more time for $x_{l,m}$ to evolve (slower learning rate) so the phase space can be more efficiently explored.

Note that the memory dynamics generate a *dynamical energy landscape* under which the voltages evolve. This guarantees that the trajectory has the ability to escape any local minima of the original, *static* energy landscape of the Boolean satisfiability problem. Visually, whenever the voltages fall into a local minimum of the original problem, the memory variables "deform" the energy landscape in such a way that the local minimum is transformed into a saddle point, and the trajectory is allowed to continue exploring the energy landscape until it finds the global minimum, which is left invariant by the memory variables (see proposition A.6.5). An extended

discussion of such dynamical properties is given in Section A.8.3.

It is advised to avoid $\gamma = 1/2$ or $\gamma = \delta$. When $C_m = 1/2$ and $\gamma = 1/2$ we find $\dot{x}_{s,m} = 0$ and the system has difficulty leaving the ambiguous state. To avoid this complication, assign $\gamma < 1/2$. Eq. (A.4) gives $x_{l,m}$ the ability to decay and it aids dynamics to have $0 < \delta < \gamma$. Assigning δ less than γ allows the system an indirect means to influence $x_{s,m}$ when $\dot{x}_{s,m} = 0$ and $C_m \neq 0$, by allowing $x_{l,m}$ to continue to grow ($\dot{x}_{s,m} = 0$ and $\dot{x}_{l,m} > 0 \implies \gamma = C_m > \delta$). The parameter $0 < \varepsilon \ll 1$ is chosen as a small positive number to guarantee that the dynamics of the short-term memory does not terminate when it reaches $x_{s,m} = 0$.

Equations (A.2)-(A.4) have been numerically integrated with the forward Euler method using an adaptive time step, $\Delta t \in [2^{-7}, 10^3]$, until all clauses have been satisfied, as determined from thresholding Eq. A.1 for all m. The code has been written in interpreted MATLAB R2019b. Each attempt at solving a clause distribution control (CDC) instance was performed on a single core (no parallelization employed) of an AMD EPYC 7401 server.

Note that the above integration scheme is the most basic and, hence, the most unstable we could implement. We thus expect more refined integration schemes may provide both better stability and scaling.

A.2.2 Trends of several indicators

In Fig. A.1, we show the typical-case behavior of other indicators in the DMM's dynamics as a function of problem size, $N \in [100, 200, ...3000]$, for difficult CDC instances, corresponding to $p_0 = 0.08$, $\alpha_r = 4.3$. Each data point is the median value of 100 instances, where 51 or more instances have been solved, with $N \leq 2600$ having 90 or more instances solved before a timeout of 10^8 steps. We observe a power-law growth ($\sim N^a$ with $a = 2.4$ in Fig. A.1(b)) in the time

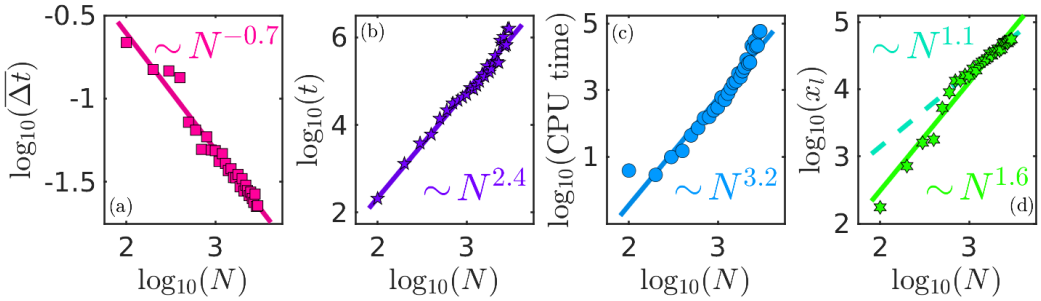

Figure A.1: Typical-case analysis of other numerical indicators for the $p_0 = 0.08$, $\alpha_r = 4.3$, CDC instances, with $N \in [100, 200, ...3000]$. Each data point is the median value of 100 instances. (a) The average time step, $\overline{\Delta t}$ (arb. units), showcases a power-law decay. (b) The median time-to-solution for the integration variable, t (arb. units), scales with power-law growth. (c) The CPU time (seconds) scales with power-law growth (d) The median of the maximum values of x_l (arb. units) when the solutions were found. There appears to be a transition in the data, so for a more informative fit (dashed line) we used data from $N \geq 10^3$, which have nearly linear growth.

variable, t (arb. units), and in the CPU time ($a = 3.2$ in Fig. A.1(c)), measured in seconds by MATLAB.

We also monitored the growth of x_l to make sure there were no exponential "energy" costs. For each instance, we collect the maximum value of x_l, then find the median of those values. Figure A.1(d) confirms that the typical growth of the maximum value of x_l follows a power law. Visually, we can see the fit ($a = 1.6$) on the data from $N \in [100, 200, ...3000]$ is poor. However, when we fit data for $N \in [1000, 1100, ..., 3000]$ the fit is almost linear ($a = 1.1$), in agreement with the approximately linear growth in Eq. (A.4) above (by taking $(C_m - \gamma)$ to be a positive constant).

Finally, we observe a power-law decay in the mean size of the time step of our adaptive integration scheme as a function of problem size (Fig. A.1(a)). In other words, as the problem size increases, the average time step is decreasing with a lower polynomial bound, rather than exponentially decaying. This observation invites modifications for speeding up solutions without introducing exponential growth into the system.

A.2.3 Trends for different values of p_0

In the generation of Barthel instances [BHL$^+$02], the parameter p_0 increases the backbone size as $p_0 \to 0.25$ (see also Sec. A.3.3). A large backbone implies, though not necessarily, a more difficult instance to solve because the solution space is smaller (less solutions). In Fig. A.2, we indeed see the exponent of the power-law scaling of the typical-case (median) CDC instances increases with increasing p_0.

The increase of backbone size also seems to cause issues with the forward-Euler integration scheme. We observe that our DMM algorithm encounters integration issues when attempting to extend these trends farther. This indicates that reducing the lower bound of the time step and/or a better integration scheme would be beneficial. Thus, we terminate simulations when the median number of steps is beyond 10^8.

To effectively sample the distribution for typical-case analysis requires a larger sample size per data point. In Fig. A.2, each data point represents the median number of steps for a sample of 500 instances.

For $p_0 = 0.08$ and $p_0 = 0.1$, $N \in [100, 200, ...1000]$. For $p_0 = 0.15$ and $p_0 = 0.2$, the forward Euler integration scheme becomes unreliable before $N = 1000$ could be reached, and such failures occur after 10^8 steps. For $p_0 = 0.15$, $N \in [100, 200, ...600]$, and for $p_0 = 0.2$, $N \in [100, 150, ...350]$. This, again, indicates that decreasing the lower bound in the time step and/or a better integration scheme is needed for large N instances. The power-law exponents calculated are $b_{0.08} = 3.0 \pm 0.3$, $b_{0.10} = 3.6 \pm 0.3$, $b_{0.15} = 5.5 \pm 0.7$, and $b_{0.20} = 6.6 \pm 1.1$. (Note that $b_{0.08} = 3.0$ differs from the value reported in the main text because we are fitting data for $N \in [100, 200, ..., 1000]$, with each data point being the median of 500 instances, rather than 100 as in the main text.) We compare these results to the 2017 Random Track competition winner,

YalSAT [Bie17], which clearly showcases exponential scaling.

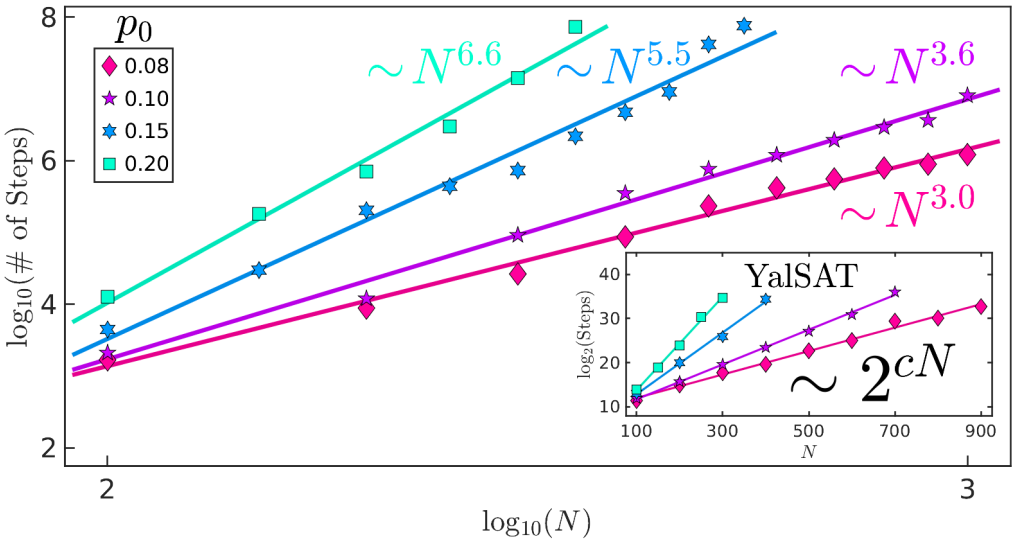

Figure A.2: Evidence for power-law scaling (aN^b) for various values of p_0, with $b_{0.08} = 3.0 \pm 0.3$, $b_{0.10} = 3.6 \pm 0.3$, $b_{0.15} = 5.5 \pm 0.7$, and $b_{0.20} = 6.6 \pm 1.1$. (inset) We use the 2017 Random Track competition winner, YalSAT [Bie17], to test scalability of a state-of-the-art algorithm. The fitted values of the exponential rates, in arbitrary units, are: $c_{0.08} = 0.03 \pm 0.002$, $c_{0.10} = 0.04 \pm 0.002$, $c_{0.15} = 0.07 \pm 0.02$, $c_{0.20} = 0.11 \pm 0.01$.

A.2.4 10-*th* to 90-*th* percentile range

Here, we show results beyond our typical-case analysis without changing any parameters or integration scheme. We find a power-law trend as a function of problem size, N, at both the 10-*th* percentile and 90-*th* percentile for $p_0 = 0.08$.

In Fig. A.3, each data point for $\alpha_r = 4.3$ is a median value of 100 instances, where $N \in [100, 200, ..., 2600]$; $\alpha_r = 5$, with $N \in [100, 200, ..., 1000,] \cup [2000, 3000, ..., 10^4]$; $\alpha_r = 6$ with $N \in [100, 200, ..., 1000] \cup [2500, 5000, 7500, 10^4, 2.5 \times 10^4, 5 \times 10^4, 10^5, 10^6]$.

Notice how the slopes of $\alpha_r = 5, 6$ appear to be converging. This may indicate that

finite-size effects contribute to the variance of solution steps. In Fig. A.3, the $\alpha_r = 6$ data points at $N = 10^6$ fall below their respective power-law trend lines. This behavior was also observed in Fig. 2 of the main text for $\alpha_r = 6, 7, 8$.

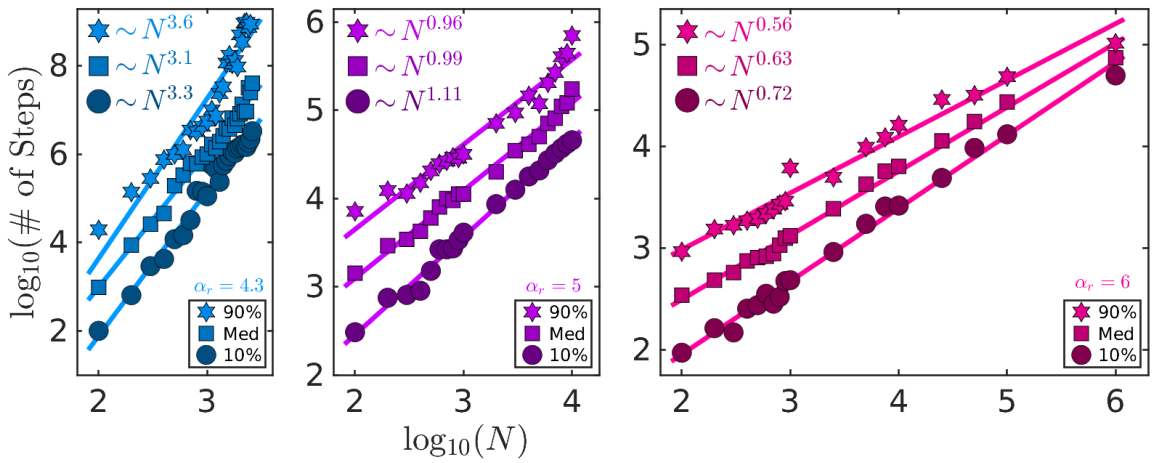

Figure A.3: Extending our typical-case analysis to include the 10-*th* and 90-*th* percentiles for $\alpha_r = 4.3, 5, 6$ and $p_0 = 0.08$ fitted to power-law trends.

A.2.5 Competition instances

We sought an independent verification of our DMMs by applying them to instances taken from previous SAT competitions [Heu17, Heu18]. Our solver was not designed for competition, so we added a heuristic to enhance its performance. Some competition instances are labeled "barthel" ($\alpha_r = 4.3$), "komb" ($\alpha_r = 5.205$), and "qhid" ($\alpha_r = 5.5$). As shown in Fig. A.4, our DMM is capable of solving all 285 competition problems from the 2017 and 2018 "Random Tracks" bearing one of these three labels. Furthermore, we can solve all of these instances within the competition's allotted CPU time (5000 second timeout). While we cannot directly compare CPU times of different machines, the reader can easily verify that our AMD EPYC 7401 server does not have any significant advantage over the machines used in the 2017 and 2018 competitions.

We chose to focus on the "small" competition instances because so many competition

solvers failed to solve them. For instance, in the 2017 Random Track there were 120 "small" instances that should be "easy" to solve in 5000 seconds. However, the 2017 Random Track winner (YalSAT) solved 124 out of 300 competition instances [Heu17]. Similarly, in the 2018 Random Track there were 165 "small" instances that should be "easy" to solve in 5000 seconds. The 2018 Random Track winner (Sparrow2Riss-2018) solved 188 out of 255 competition instances [Heu18]. With the addition of more heuristics to our system, our DMM algorithm could possibly surpass previous competition performances.

We modified our algorithm to perform in the context of competition, by making one major modification: each constraint has its own α_m associated with C_m, and it is modified in regular intervals during the solution search. Initially, for all clauses, $\alpha_m = 5$, and all other parameters remain unchanged from the main text. The search for the solution is initialized as before, but after 10^4 arbitrary time units the simulation is paused to modify the values of α_m. The procedure starts by finding the median of the $x_{l,m}$ values for all m. If $x_{l,m}$ is greater than the median, then the corresponding α_m is increased by a multiplicative factor of 1.1, otherwise, the corresponding α_m is decreased by a multiplicative factor of 0.9. To prevent decay to zero, $\alpha_m = 1$ is the minimum value. If $x_{l,m}$ grows to its maximum cutoff, the process restarts by setting $x_{l,m} = 1$ and $\alpha_m = 1$. The integration is resumed without modification to any other variables or parameters, and will repeat after another 10^4 arbitrary time units.

A.2.6 Random 3-SAT

While we have chosen to use planted-solution 3-SAT instances for the stated reasons (solution existence known), other authors [BMZ05, MKHT20] choose to work with 3-SAT instances that lack clause distribution control and have no guarantee of the existence of a solution. Recall that the algorithms discussed herein are all incomplete SAT solvers, meaning they cannot prove a solution does not exist (UNSAT). Therefore, scalability tests on general random 3-SAT instances

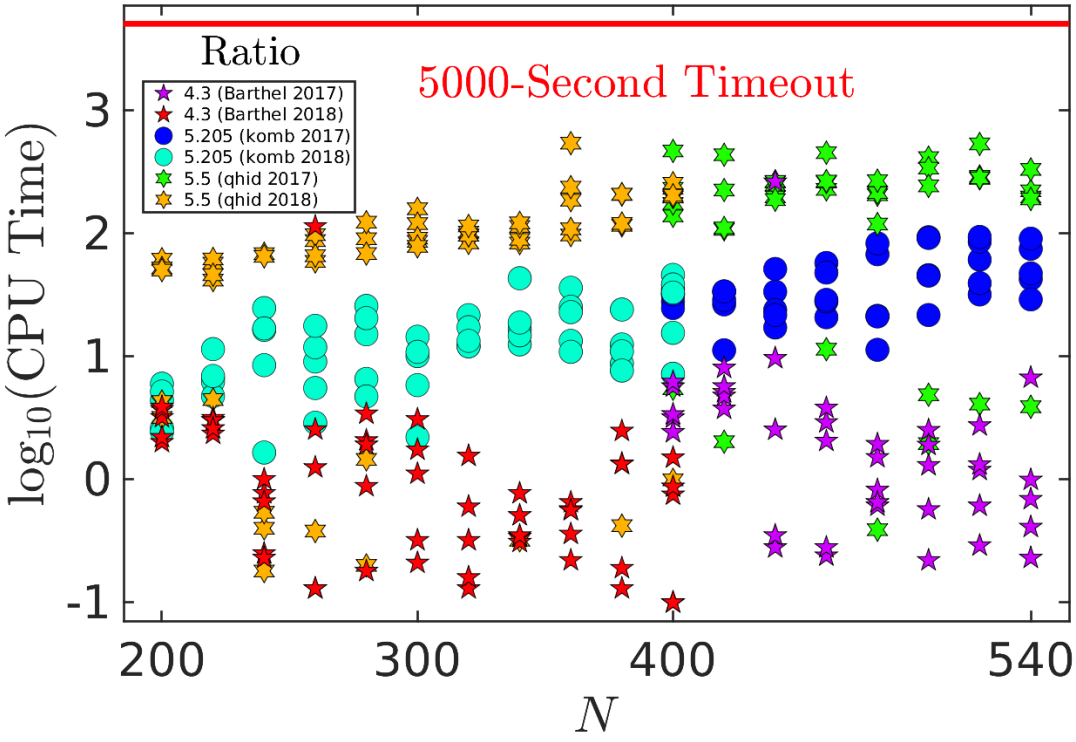

Figure A.4: Results of a DMM algorithm solving competition instances (from the 2017 and 2018 SAT competitions [Heu17, Heu18]), on a single core (no parallelization employed) of an AMD EPYC 7401 server, with only one set of random initial conditions. Note that some data points overlap.

have a degree of uncertainty regarding whether it is possible to find solutions to all instances tested. The SID algorithm removes much of the uncertainty by manipulating a property of the SAT/UNSAT transition: for $\alpha_r < 4.267$, the probability that a randomly generated instance has a solution approaches 1 as N grows [BMZ05].

When N is small, it is unlikely all generated instances will be satisfiable, so the numerical simulation of AnalogSAT [MKHT20] takes another approach to generate satisfiable instances. Starting with random 3-SAT instances, the authors use another algorithm, MiniSAT [ES03], to filter the instances. That is, AnalogSAT is only tested on instances that MiniSAT can solve. However, this has the drawback of excluding 3-SAT instances that the filtering algorithm is incapable of solving.

In Fig. A.5, our DMM solves all of the 3-SAT instances from Ref. [MKHT20] that have 100 instances per value of N. (Large N instances only have 1 instance per value of N.) We use the same DMM and parameters as presented in the main text, where $\zeta = 10^{-1}$ for $\alpha_r = 3.4, 3.8$, and $\zeta = 10^{-2}$ for $\alpha_r = 4.25$

The authors of Ref. [MKHT20] prefer wall time as the indicator used to show polynomial scaling, claiming it is a realistic measure of hardware. Therefore, we show scaling of both steps (Fig. A.5(a)-(c)) and wall time (Fig. A.5(d)-(f)).

For very small values of N, our DMM encounters overhead that dominates the wall time scalability (solution wall time $\sim 10^{-1}$ seconds). The initialization of the MATLAB code dominates the scalability for wall time so we exclude small values of N from the curve fitting procedure. (Comparing the scaling of steps and wall time in Fig. A.5, it can be seen there is no initialization effect in the scaling of steps.) With these considerations taken into account, we show several improvements.

In Fig. A.5(d), for $\alpha_r = 3.4$, we see the DMM's scalability of the maximum solution

times, $\sim N^{1.25}$, is approximately the same as that reported for AnalogSAT's scaling of the mean, $\sim N^{1.26}$ [MKHT20]. In Fig. A.5(e), for $\alpha_r = 3.8$, we see the DMM's scalability of the maximum solution times, $\sim N^{1.11}$, is better than that reported for AnalogSAT's scaling of the mean, $\sim N^{1.63}$ [MKHT20]. Additionally, our range of N goes beyond $N = 10^4$. In Fig. A.5(f), for $\alpha_r = 4.25$, we see the DMM's scalability of the maximum solution times, $\sim N^{3.55}$, is better than that reported for AnalogSAT's scaling of the mean, $\sim N^{4.35}$ [MKHT20]. For the largest value of N tested, $N = 463$, the maximum wall time is 177 seconds, where AnalogSAT's mean wall time for the same value of N is $\sim 10^3$ seconds.

For another test, we generated random 3-SAT instances at $\alpha_r = 4.25$, where no solution has been planted (0-hidden). Due to being to the left of the SAT/UNSAT transition, there is a high probability that a randomly generated 3-SAT instance will be satisfiable. Therefore, we should be able to solve more than 50% of instances generated, and can use the median as another measure of scalability. We use the same DMM and parameters as presented in the main text, with $\zeta = 10^{-2}$. In Fig. A.6, we find power-law scalability for these instances as well.

A.3 Continuous 3-SAT

In this Section, we establish the formalism for studying the continuous version of the 3-SAT problem we have solved in the main text. This continuous version generates an energy landscape that we explore with the memcomputing dynamics (see Section A.6). In addition, we provide a brief discussion on the class of planted 3-SAT instances [BHL$^+$02] that we used in this paper as benchmarks. To facilitate the theoretical analysis we will also slightly change the notation so that we can write Eqs. (A.1)-(A.6) in a more compact way.

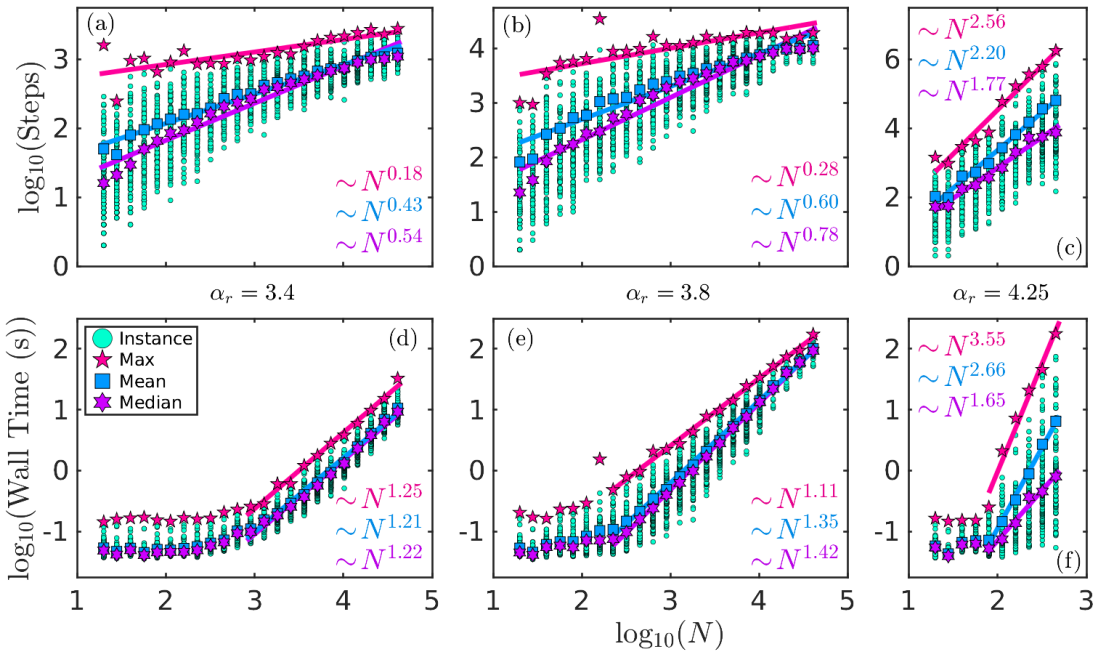

Figure A.5: Scalability on instances from Ref. [MKHT20] for $\alpha_r = 3.4$ (a) and (d), $\alpha_r = 3.8$ (b) and (e), $\alpha_r = 4.25$ (c) and (f). We show scalability in integration steps (a)-(c) and wall time (d)-(f). For each N there are 100 solved instances shown in each panel.

A.3.1 From Discrete to Continuous Variables

Consider a 3-SAT Boolean formula with n variables and m clauses, where α_r is commonly referred to as the *clause density*, as it is the ratio between the number of clauses and number of Boolean variables[1]. We let $+1$ correspond to the true assignment of a Boolean variable, and -1 to the false assignment. We then map the n Boolean variables into n continuous variables, $\mathbf{v} \in [-1,+1]^n$, which we term *voltages*. For each clause, we can define various energy functions indicating the state of satisfaction of each clause given a voltage assignment. The expression of these functions are most compactly expressed by making use of the definition of *polarity*.

Definition A.3.1 (Polarity and Constraint)**.** *Consider a 3-SAT Boolean formula with n Boolean variables and m clauses. We denote the i-th Boolean variable as x_i, and its polarity in the j-th*

[1]Note that we are using a slightly different notational conventions from the main text. In the following, n and m are cardinal numbers denoting the numbers of variables and clauses respectively, and i and j are used as their respective indices.

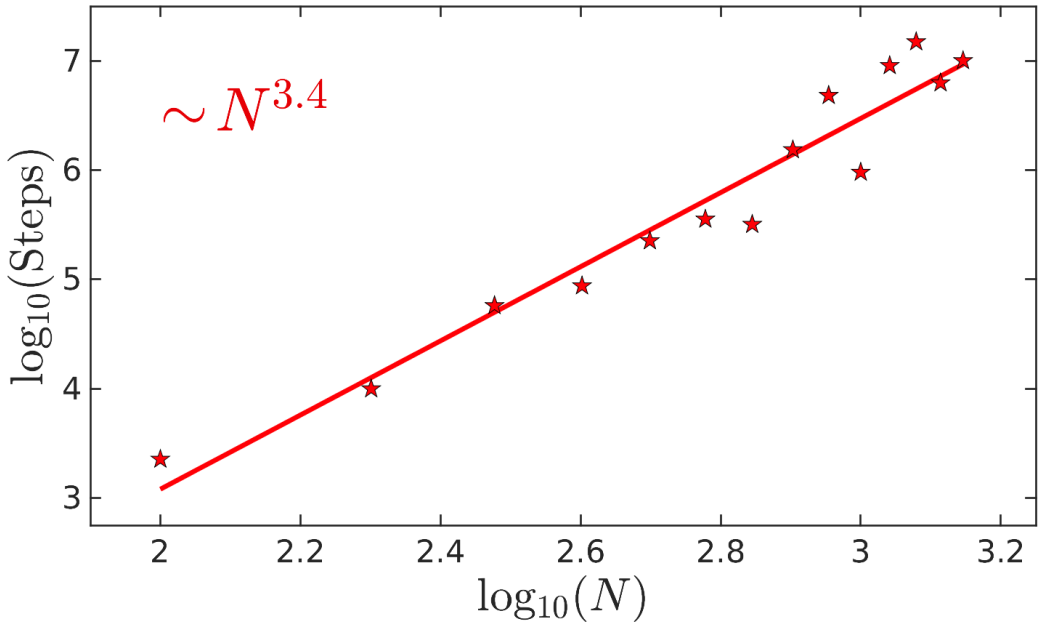

Figure A.6: Scalability of typical general 3-SAT instances at $\alpha_r = 4.25$, generated without knowledge of solution existence. For each N, we attempt to solve 100 general 3-SAT instances, and calculate the median when 51 instances have solved.

clause as

$$q_{ij} = \begin{cases} +1 & \text{if } x_i \text{ appears positively in clause } j, \\ -1 & \text{if } x_i \text{ appears negatively in clause } j, \\ 0 & \text{if } x_i \text{ does not appear in clause } j. \end{cases}$$

The polarity matrix, Q, is the matrix with the element on the i-th row and j-th column being q_{ij}. Note that a 3-SAT Boolean formula is completely specified by Q.

Given a voltage assignment $\mathbf{v} \in [-1,+1]^n$*, we rewrite Eq. (A.1), the constraint of the j-th clause as*

$$C_j(\mathbf{v}) = \frac{1}{2} \min_{\{i|q_{ij}\neq 0\}} \left(1 - q_{ij} v_i\right). \tag{A.7}$$

The global constraint is the sum of the constraints of all clauses

$$C(\mathbf{v}) = \sum_j C_j(\mathbf{v}). \tag{A.8}$$

For any $\mathbf{v}_0$ *such that* $C(\mathbf{v}_0) = 0$ *is satisfied, we call* $\mathbf{v}_0$ *a solution vector.*

Remark. $\mathbf{v}_0$ *is called a solution vector because if we take the corresponding Boolean vector* $\mathbf{x}_0$ *by thresholding* $\mathbf{v}_0$ *(converting* $v_{0,i} > 0$ *to true and* $v_{0,i} \leq 0$ *to false), then* $\mathbf{x}_0$ *must be a solution to the original 3-SAT problem. This is because the global energy being zero,* $C(\mathbf{v}_0) = 0$*, implies that every clause energy must also be zero,* $C_j(\mathbf{v}_0) = 0$*, which further implies that every clause is satisfied under the assignment* $\mathbf{x}_0$*. Note that the converse is also true; if* $C(\mathbf{v}') = 0$*, then* $\mathbf{v}'$ *must be a solution vector.*

To ease the burden of notation, it is useful to define the following index notation

$$\sigma_j = \text{argmin}_{\{i \mid q_{ij} \neq 0\}}(1 - q_{ij} v_i), \tag{A.9}$$

which can be simply interpreted as the index of the voltage whose assignment is closest to satisfaction among all voltages in clause j. Note that by this definition, we have $q_{\sigma_j, j} = \pm 1$, denoting the polarity of the Boolean variable whose assignment determines the value of $C_j(\mathbf{v})$. This notation allows us to simplify the expression of the clause constraint as given in Eq. (A.7)

$$C_j(\mathbf{v}) = \frac{1}{2}(1 - q_{\sigma_j, j} v_{\sigma_j}).$$

Note that if the goal is for an effective numerical implementation of the memory dynamics solely as a means to find a solution, rather than relaxing into an equilibrium point, one can exploit

the fact that if an assignment of $\mathbf{v}$ such that $C_j(\mathbf{v}) < \frac{1}{2}$ for every clause, then the original 3-SAT problem is solved by thresholding $\mathbf{v}$ to generate $\mathbf{x_0}$.

Proposition A.3.1. *Given an assignment of the voltages* $\mathbf{v} \in [-1,+1]^n$ *such that* $\mathbf{C}(\mathbf{v}) < \frac{1}{2}$, $\mathrm{sgn}(\mathbf{v})$ *is a solution vector*[2].

Proof. Recall that $C_j(\mathbf{v}) = \frac{1}{2}(1 - q_{\sigma_j j} v_i)$. Since $\forall j$ we have $C_j(\mathbf{v}) < \frac{1}{2}$, then $q_{\sigma_j j} v_{\sigma_j} > 0$. If we let $\mathbf{v}_0 = \mathrm{sgn}(\mathbf{v})$, then $q_{\sigma_j j} v_{0,\sigma_j} = q_{\sigma_j j}\,\mathrm{sgn}(v_{\sigma_j}) = \mathrm{sgn}(q_{\sigma_j j} v_{\sigma_j}) = +1$, as $q_{\sigma_j j} = \pm 1$. Therefore, we have $C_j(\mathbf{v}_0) = \frac{1}{2}(1 - q_{\sigma_j j} v_{0,\sigma_j}) = \frac{1}{2}(1-1) = 0$, so $\mathcal{C} = \sum_j C_j(\mathbf{v_0}) = 0$. Therefore, $\mathrm{sgn}(\mathbf{v})$ is a solution vector. □

Remark. *This means that once we have discovered an assignment of voltages such that the constraints of all clauses are less than* $\frac{1}{2}$*, we can simply threshold the voltages to obtain the corresponding Boolean variables for a solution of the original 3-SAT problem.*

The global constraint defined in Eq. (A.8) is not everywhere differentiable with respect to the voltages due to the use of a minimum operation, and this causes some inconvenience in analyzing certain properties of the 3-SAT problem structure from the perspective of statistical mechanics (see Eq. (A.15)). We then construct an energy function that is continuous (and also smooth) in anticipation of such analysis.

Definition A.3.2 (Energy)**.** *Given a 3-SAT Boolean formula defined by an* $n \times m$ *polarity matrix* Q*, we define the energy of the* j*-th clause for any voltage assignment* $\mathbf{v} \in [-1,+1]^n$ *as*

$$E_j(\mathbf{v}) = \frac{1}{8} \prod_{\{i \,|\, q_{ij} \neq 0\}} (1 - q_{ij} v_i). \tag{A.10}$$

[2]While it is possible for $v_i = 0$, resulting in $\mathrm{sgn}(0) = 0$, this rare event does not affect the remaining nonzero voltages from satisfying all clauses. In such a scenario, x_i can be set to TRUE or FALSE without affecting the satisfiability of the solution vector.

The global energy is the sum of the energies for all clauses

$$\mathcal{E}(\mathbf{v}) = \sum_j E_j(\mathbf{v}). \tag{A.11}$$

Remark. *We can show in a similar fashion (see the remark of definition A.3.1) that if the 3-SAT problem is satisfiable, then the global energy of a solution vector* $\mathbf{v}_0$ *will also be zero, or* $\mathcal{E}(\mathbf{v}_0) = 0$*, which is also its global minimum. The converse is also true. Therefore, the problem of minimizing the global constraint,* $\mathcal{C}$*, and minimizing the global energy,* $\mathcal{E}$*, are in fact equivalent problems.*

The flow field of the memory dynamics for the voltages (see Section A.6) contains two terms, one being similar to the gradient of $\mathcal{E}(\mathbf{v})$ (see Eq. (A.18)) which we name the *gradient-like term* and the other one closely related the clause function $C_j(\mathbf{v})$ (see Eq. (A.19)) which we name the *rigidity term*. At certain hyperplanes, the gradient-like term is not differentiable and the rigidity term is discontinuous (see section A.6.1). We develop the mathematical formalism for studying such irregular flow fields in Section A.5.

A.3.2 Gauging the 3-SAT Problem

If the original 3-SAT Boolean formula has a known solution, analysis can be simplified by converting the 3-SAT formula into an equivalent 3-SAT formula in such a way that the known solution of the original formula is now a solution to the gauged formula with an all-true assignment of the Boolean variables[3]. After the conversion, there will be a restriction on the possible clause types that can appear in the formula (no clause appears with all variables negated). This will allow for a natural description of the clause distribution control (CDC) class of planted instances, and greatly simplify the analysis of memory dynamics.

[3]For all planted-solution CDC instances generated for numerical simulations, the all-true solution is first assumed and then randomly changed by a local gauge transformation [BHL$^+$02] to remove any solver bias towards the all-true solution.

Definition A.3.3 (Gauge Fixing)**.** *Consider a satisfiable 3-SAT Boolean formula given by an $n \times m$ polarity matrix Q. Given any solution $\mathbf{x}_0$ to the 3-SAT problem, we gauge fix the polarity matrix Q with respect to $\mathbf{x}_0$, $G_{\mathbf{x}_0} : \{-1,+1\}^{nm} \mapsto \{-1,+1\}^{nm}$, such that each element of Q transforms as follows*

$$q_{ij} \mapsto x_{0,i} q_{ij}.$$

We refer to $Q' = G_{\mathbf{x}_0}(Q)$ as the gauged polarity matrix.

Remark. *It can be easily shown that the formulas given by Q and Q' have the same structure. In particular, given some mapping of the polarity matrix $G_{\mathbf{x}_0}$, we can simultaneously map each Boolean state $\mathbf{x}$ to a new one as follow*

$$\mathbf{x} \mapsto \mathbf{x} * \mathbf{x}_0,$$

where $*$ denotes component-wise multiplication. It is then obvious that the satisfaction state of each literal $q_{ij}x_i$ is invariant under this mapping. A similar procedure applies for mapping the voltages as well

$$\mathbf{v} \mapsto \mathbf{v} * \mathbf{v}_0.$$

Note that the performance of most SAT solvers (including the canonical Walk-SAT algorithm [SK93] and our memcomputing one as presented in Eqs. (A.1)-(A.6)) are invariant under gauge conjugation [MM09]. Informally, this means that nothing is gained or lost in terms of the efficiency of optimization by gauging the problem first before running the algorithm, as the behavior of a SAT solver at each time step will not change under a gauge mapping (see Section A.6.3). The choice to gauge fix a solution to $+\mathbf{1}$ is purely for analytic convenience.

An important property of a gauge fixed 3-SAT formula is that no clause can contain three negated Boolean variables.

Lemma A.3.2. *Given a gauged polarity matrix Q of a k-SAT problem [GJ90], we have the following*

$$\forall j, \exists i, q_{ij} = +1.$$

In other words, all clauses must contain at least one literal that is an unnegated variable.

Proof. We prove this by contradiction. We first assume the negation of the Lemma, then

$$\exists j, \forall i, q_{ij} = -1.$$

Then without loss of generality (WLOG), we can assume that the j-th clause is the following

$$(\bar{x}_1 \vee \bar{x}_2 \vee ... \vee \bar{x}_k).$$

Since Q is a gauged polarity matrix, a solution must be $\mathbf{x}_0 = +\mathbf{1}$. However, this assignment evaluates to false by the above clause, so it cannot be a solution. We therefore have a contradiction.

□

Remark. *It should be noted that the inclusion of clauses with all negations does not preclude the possibility of the formula being satisfiable, as solutions other than $+\mathbf{1}$ may still exist.*

Lastly, we point out that the clause constraint defined in Eq. (A.7) has the important property of being invariant under a gauge mapping.

Lemma A.3.3 (Gauge Invariance of Constraints)**.** *Given a satisfiable 3-SAT instance with some solution vector $\mathbf{v}_0$, $C_j(\mathbf{v})$ is invariant under the following transformation for $\forall j$*

$$q_{ij} \mapsto q_{ij} v_{0,i} \qquad \mathbf{v} \mapsto \mathbf{v} * \mathbf{v}_0.$$

Proof. Recall from Eq. (A.7) that

$$C_j(\mathbf{v}) = \frac{1}{2} \min_{q_{ij} \neq 0} \left(1 - q_{ij} v_i\right).$$

If we let $q'_{ij} = q_{ij} v_{0,i}$ and $v'_i = v_i v_{0,i}$, then we have

$$C'_j = \frac{1}{2} \min_{q'_{ij} \neq 0} \left(1 - q'_{ij} v'_i\right) = \frac{1}{2} \min_{q_{ij} \neq 0} \left(1 - q_{ij} v_i (v_{0,i})^2\right) = C_j,$$

where we note that $v_{0,i} = \pm 1$, so $(v_{0,i})^2 = 1$. □

Remark. *It directly follows that the global constraint must be gauge invariant as well. It can be shown in a similar fashion that the energy of each clause is also gauge invariant.*

A.3.3 Planted Instances

Here, we consider a class of random 3-SAT instances generated with a planted solution to guarantee an instance to be satisfiable, however, planted in such a way so as to be hard for local-search SAT solvers to find [BHL$^+$02]. In particular, we consider instances whose polarity matrix Q satisfies Lemma A.3.2 up to a gauge mapping. In other words, when we construct Q, we cannot allow the appearance of columns whose nonzero elements are all -1. We formally describe a particular method of constructing such matrices in the following section.

Randomly Planted Formula

We first consider the general method of generating satisfiable formulas where every clause is formed independently by randomly including Boolean variables, with the clause type randomly sampled from some given distribution [HR04].

Definition A.3.4 (Planted Instance)**.** *We consider a random matrix Q generated by parameters*

$\{\alpha_r, p_0, p_1, p_2\}$ *that satisfies the following normalization condition*

$$p_0 + 3p_1 + 3p_2 = 1. \tag{A.12}$$

For each column j, we randomly select three distinct rows $\{i_{j,1}, i_{j,2}, i_{j,3}\}$ uniformly. We then randomly assign the elements $(q_{i_{j,1}j}, q_{i_{j,2}j}, q_{i_{j,3}j})$ with an element from the following set

$$\{(q_1, q_2, q_3) \in \mathbb{R}^3 \;\big|\; |q_1| = |q_2| = |q_3| = +1\} \big/ (-1, -1, -1),$$

with each assignment associated with the sampling probability given as follows

$$p_0 : q_1 + q_2 + q_3 = 3,$$
$$p_1 : q_1 + q_2 + q_3 = 1,$$
$$p_2 : q_1 + q_2 + q_3 = -1.$$

We then assign all other elements in column j to zero.

Remark. *To explain this construction in simple terms, we can consider a 3-SAT Boolean formula where each clause is independently generated through the inclusion of 3 randomly chosen Boolean variables out of the n total variables without replacement. The negations of the Boolean variables in the clause are randomly assigned such that there is a probability p_0 that all variables appear without negation; there is a probability $3p_1$ that only one variable is negated (the prefactor of 3 is to account for the fact that there are 3 possible variables to negate); and there is a probability $3p_2$ that two variables are negated (the prefactor of 3 arises similarly).*

Clause Distribution Control Instances

We now consider a class of hard instances [BHL$^+$02] that is generated based on the method described in Definition A.3.4. In particular, the generation method is restricted in the

presence of a new constraints on the parameters $\{\alpha_r, p_0, p_1, p_2\}$, in addition to the normalization condition given in Eq. (A.12). This gives us only $4-2=2$ degrees of freedom in the selection of the parameters, α_r and p_0.

Definition A.3.5 (Clause Distribution Control Instances)**.** *A Clause Distribution Control [4] (CDC) instance generated with the parameters α_r and p_0 is an instance whose polarity matrix Q is randomly generated by the following constraints*

$$\alpha_r > 4.25, \qquad 0.077 < p_0 < 0.25, \qquad p_1 = \frac{1-4p_0}{6}, \qquad p_2 = \frac{1+2p_0}{6}, \tag{A.13}$$

based on the method given in Definition A.3.4.

Remark. *It has been claimed that this class of instances is difficult for local-search procedures [BHL$^+$02], though, it has been shown that the difficulty does not persist for some upper limit on α_r that depends on the problem size, n [BS15].The results from the Walk-SAT algorithm confirm the instances generated for numerical simulation are difficult in that the showcase exponential scalability.*

The reason for enforcing the condition $p_0 < \frac{1}{4}$ is twofold. First, p_0 is restricted so that parameter p_1 is non-negative, as it represents a probability. Second, the instances created with $p_0 = 1/4$ are known to be solvable in polynomial time using a global algorithm [BHL$^+$02]. It can be easily verified that the probabilities given in Eq. (A.13) satisfy the normalization condition (Eq. (A.12)) in addition to the following condition

$$p_0 + p_1 - p_2 = 0 \tag{A.14}$$

If the above constraint is satisfied, then it can be shown that a greedy local-search SAT solver

[4] While the method can be generalized, for example as in Ref. [JMS07], we report the method outlined in Ref. [BHL$^+$02].

initialized with a random assignment of variables will not be biased towards the planted solution [BHL+02]. In the language of statistical mechanics, we say that the instance is equivalent to an instance of a disordered diluted spin glass with couplings up to three spins [MZ97]. The Hamiltonian of this diluted spin glass can be written as

$$H = -\sum_{i} H_i S_i - \sum_{ij} T_{ij} S_i S_j - \sum_{ijk} S_i S_j S_k, \tag{A.15}$$

which is equivalent to the global energy as defined in Eq. (A.11). If Eq. (A.14) is enforced, then the average of the local field over the disorder $\overline{H}_i$ is zero for all spins, so there is typically no direct bias towards the planted state $\mathbf{S} = +\mathbf{1}$. An extended discussion of the CDC instances can be found in literature on the statistical mechanics of Boolean satisfiability problems [HR04].

Solution Backbone and Cluster

As briefly addressed in the remark of Lemma A.3.2, planting the $+\mathbf{1}$ solution in an instance does not forbid the existence of additional solutions. In fact, multiple solutions may exist, however, their locations in phase space, with respect to one another, and the similarity of solutions are generally what determine the difficultly of an instance. In most cases, some solutions will overlap non-trivially, meaning that their assignments will coincide for a certain number of variables. For instances admitting overlapping solutions, there are two concepts (occurring non-exclusively) important for analytic studies.

For the first concept, given a solution to an instance, we can define a solution *cluster* as the subset of all solutions that can be assigned from the given solution via a sequence of single spin flips (Boolean variable negation) [ERT11]. Note, after each flip the assignment must remain a solution to be considered part of the cluster. While the clustering of solutions into one big cluster may intuitively seem like a more difficult instance, knowing only one solution cluster

exists is not enough information to categorize an instance as more difficult than others. The second concept will give additional information about the difficulty. Given the set of all solutions, we define the *backbone* to be the number of variables that appear with only one parity in all solutions [HR04]. In other words, for the SAT solver to find a solution, it is necessary for the backbone to be assigned correctly[5]. In general, the emergence of a backbone in a 3-SAT instance results in variables that must be assigned to a particular value to find any solution (an inherent difficulty), however, there can still exist a local field that can guide a greedy local-search SAT solver to the solution.

To understand why the CDC instances (planted solution) are difficult, it aids understanding to describe the solution cluster distribution in uniform random 3-SAT (no guaranteed solution). Using the replica symmetry approximation [MZ96, MZ97, MZK$^+$99], a variational approach accounting for replica-symmetry breaking [BMW00], and the cavity method [MPZ02, MZ02] from statistical mechanics, it was shown that the 3-SAT problem undergoes phase transitions as clause density is increased [6]. For $\alpha_r < \alpha_d \simeq 3.92$, there is one large solution cluster, and solutions are relatively easy to find. At α_d the large solution cluster breaks into an exponential amount of solution clusters, with an exponential amount of solutions within each. These clusters are far from each other in phase space, and their frequency diminishes as $\alpha_r \to \alpha_c \simeq 4.267$, until only one solution cluster remains. That is, the solutions become less frequent as α_c (the complexity peak) is approached, until no solutions exist (the SAT/UNSAT transition) [HR04].

At $p_0 = 0.25$, for $\alpha_r < 4.27$, there is no difference between the CDC class and uniform random 3-SAT, with the solution entropy and clustering transition, α_d, being the same [HR04]. However, the SAT/UNSAT transition at $\alpha_c \simeq 4.27$ is obviously absent, being that the solution is

[5] In the case of the CDC instances that we use, the fashion in which the backbone appears as the clause density is increased is dictated directly by the parameter p_0. More particularly, this CDC parameter induces a phase transition from a continuous appearance of a backbone to a discontinuous appearance of a backbone [BHL$^+$02].

[6] See Ch. 7 of Ref. [HR04] for a self-contained account of the following results.

always planted. Now, the instance class undergoes a first-order ferromagnetic transition at α_c, resulting in only one solution cluster remaining. The first-order transition is more pronounced for $0.077 < p_0 < 0.25$, and there is a discontinuous appearance of a backbone. (For $p_0 < 0.077$, no backbone appears.) At $\alpha_c \simeq 4.27$, the paramagnetic phase (many solution clusters) transitions to a ferromagnetic phase (one cluster containing the planted solution) with the discontinuous appearance of a backbone[7].

The approximate backbone size for CDC instances range from $0.72n$ at $p_0 \simeq 0.077$ to $0.94n$ at $p_0 = 0.25$ [BHL$^+$02]. Therefore, with all factors considered above, p_0 serves as a measure of difficulty for the CDC instances.

In this material, we base our focus on the study of the dynamical properties of our DMM by defining solution planes on hyperfaces of the voltage hypercube [ERT11]. When the solution vector is on a hyperface that corresponds to a solution plane (see A.6.4), the voltage dynamics are near a branch of a solution cluster, effectively solving the CDC instance. To further associate the concepts, when a solution is found on a vertex of the hypercube, the solution cluster can be traversed by traveling along the hyperedges of the hypercube that connect to other solution vertices.

A.4 Lipschitz Continuity

Before we present the equations governing the dynamics of our memcomputing solver in Section A.6, it is necessary to first introduce a few formal mathematical arguments that will help establish the existence and uniqueness of the solution trajectory under an ordinary differential equation (ODE). For instance, the requirement for the existence and uniqueness of a local solution

[7]The reader may notice the transition is reported as $\alpha_c \simeq 4.27$, but Def. A.13 has $\alpha_r > 4.25$. To avoid any discrepancy, the smallest ratio used in numerical simulations is $\alpha_r = 4.3$.

to a first order autonomous ODE is the Lipschitz continuity of the flow field [Ver06]. We begin by formally defining Lipschitz continuity.

Definition A.4.1 (Lipscthiz Continuity)**.** *Let X and Y be two metric spaces. A function $f : X \mapsto Y$ is Lipschitz continuous if there is a real constant $K \geq 0$ such that*

$$\forall x_1, x_2 \in X, \quad d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2),$$

where d_X and d_Y denote the metrics on X and Y respectively.

Remark. *This definition can be easily specialized to a vector field $V : \mathbb{R}^n \mapsto \mathbb{R}^n$.*

Theorem A.4.1 (Picard–Lindelöf theorem)**.** *Given a Lipschitz continuous vector field $V : \mathbb{R}^n \mapsto \mathbb{R}^n$, the classical solution $\mathbf{x}(\mathbf{x_0}, t)$ to the first order autonomous ODE, $\dot{\mathbf{x}}(t) = V(\mathbf{x})$, exists and is unique for $\forall t \in \mathbb{R}$.*

Our dynamics are governed by a high dimensional vector flow field, $F : \mathbb{R}^n \mapsto \mathbb{R}^n$. To study the Lipschitz continuity of the vector field F, we simply study the Lipschitz continuity of the field components in the quotient spaces instead, by the following lemma.

Lemma A.4.2. *Given a metric space X, and a product metric space $Y = Y_1 \times Y_2 \times ... \times Y_n$ equipped with a p-product metric, where $p \in (0, +\infty)$, let $f_i : X \mapsto Y_i$ be a mapping and $f : X \mapsto Y$ be defined as $f(x) = \big(f_1(x), f_2(x), ..., f_n(x)\big)$. Then f is Lipschitz continuous if and only if f_i is Lipschitz continuous for $\forall i \in [[1, n]]$.*

Proof. We first assume that f_i is Lipschitz continuous $\forall i$, with its Lipschitz constant being K_i.

Then $\forall x_1, x_2 \in X$, we have

$$\begin{aligned} d_Y\big(f(x_1), f(x_2)\big) &= \Big(\sum_{i=1}^{n} d_{Y_i}\big(f_i(x_1), f_i(x_2)\big)^p \Big)^{1/p} \\ &\leq \Big(\sum_{i=1}^{n} K_i^p \, d_X(x_1, x_2)^p \Big)^{1/p} \\ &\leq \Big(\max_i (K_i)^p \sum_{i=1}^{n} d_X(x_1, x_2)^p \Big)^{1/p} \\ &= \big[\max_i (K_i) n^{1/p} \big] d_X(x_1, x_2). \end{aligned}$$

In other words, the Lipschitz constant for f is simply $\max_i(K_i)n^{1/p}$ so f is Lipschitz continuous.

Now, we assume that $f_{i'}$ is not Lipschitz continuous for some i'. Then $\exists x_1, x_2 \in X$ such that

$$\forall K \geq 0, \quad d_{Y_i'}\big(f_{i'}(x_1), f_{i'}(x_2)\big) > K d_X(x_1, x_2).$$

We then have

$$\begin{aligned} d_Y\big(f(x_1), f(x_2)\big) &= \Big(\sum_{i=1}^{n} d_{Y_i}\big(f_i(x_1), f_i(x_2)\big)^p \Big)^{1/p} \\ &\geq \Big(d_{Y_{i'}}\big(f_{i'}(x_1), f_{i'}(x_2)\big)^p \Big)^{1/p} \\ &> K d_X(x_1, x_2), \end{aligned}$$

meaning that f is also not Lipschitz continuous. □

For our work, we are also interested in the Lipschitz continuity of a vector field that is projected onto another vector field. In particular, in definition A.5.7, we show how a vector field can be projected onto a regular surface. In the following lemma, we give the condition for this "projected" vector field to be Lipschitz continuous. From here on, we shall use the notation $\langle a, b \rangle$ to denote the inner product of vectors a and b.

Lemma A.4.3 (Continuity of Projection)**.** *Let* $\text{proj}_{\mathbf{v}} : \mathbb{R}^n \mapsto \mathbb{R}^n$ *be the projection mapping defined as*

$$\text{proj}_{\mathbf{v}}(\mathbf{u}) = \langle \mathbf{u}, \mathbf{v} \rangle \frac{\mathbf{v}}{||\mathbf{v}||^2} = \langle \mathbf{u}, \hat{\mathbf{v}} \rangle \hat{\mathbf{v}}.$$

Let X be a metric space. Let $f_1 : X \mapsto \mathbb{R}^n$ *be some Lipschitz continuous function bounded from below by* $\exists m > 0$ *in norm, and let* $f_2 : X \mapsto \mathbb{R}^n$ *be some Lipschitz continuous function bounded from above by* $\exists M > 0$*. Then* $f(x) = \text{proj}_{f_1(x)}\big(f_2(x)\big)$ *is Lipschitz continuous.*

Proof. $\forall x_1, x_2 \in X$, we have

$$||f_1(x_2) - f_1(x_1)|| \leq K_1 d(x_1, x_2); \qquad ||f_2(x_2) - f_2(x_1)|| \leq K_2 d(x_1, x_2),$$

for some constants $K_1, K_2 > 0$. For the sake of simplicity, we denote $f_1 = f_1(x_1)$, $f_2 = f_2(x_1)$, $f_1' = f_1(x_2)$, and $f_2' = f_2(x_2)$. Then we can write

$$\begin{aligned} ||f(x_2) - f(x_1)|| &= ||\text{proj}_{f_1'}(f_2') - \text{proj}_{f_1}(f_2)|| \\ &= \Big|\Big| \langle f_2', \hat{f}_1' \rangle \hat{f}_1' - \langle f_2, \hat{f}_1 \rangle \hat{f}_1 \Big|\Big| \\ &= \Big|\Big| \langle f_2' - f_2, \hat{f}_1 \rangle \hat{f}_1 + \big(\langle f_2', \hat{f}_1' \rangle \hat{f}_1' - \langle f_2', \hat{f}_1 \rangle \hat{f}_1 \big) \Big|\Big| \\ &\leq \Big|\Big| \langle f_2' - f_2, \hat{f}_1 \rangle \Big|\Big| + \Big|\Big| \big(\langle f_2', \hat{f}_1' \rangle \hat{f}_1' - \langle f_2', \hat{f}_1 \rangle \hat{f}_1 \big) \Big|\Big|. \end{aligned} \tag{A.16}$$

Note that the first term is bounded as follows

$$\Big|\Big| \langle f_2' - f_2, \hat{f}_1 \rangle \Big|\Big| \leq \Big|\Big| f_2' - f_2 \Big|\Big| \Big|\Big| \hat{f}_1 \Big|\Big| \leq K_2 d(x_1, x_2).$$

To bound the second term, it is convenient to denote $\phi = \arccos\big(\langle \hat{f}_1, \hat{f}_1' \rangle \big)$, then it can be easily shown that

$$\phi \leq \begin{cases} \arcsin\big(\frac{K_1 d(x_1, x_2)}{m}\big) & \text{if} \quad K_1 d(x_1, x_2) \leq m, \\ \pi & \text{otherwise.} \end{cases}$$

This means that $\phi \leq \frac{2K_1 d(x_1,x_2)}{m}$. We then see that the second term in the last line of Eq. (A.16) is bounded as follows

$$\left|\left|\left(\langle f_2', \hat{f}_1'\rangle \hat{f}_1' - \langle f_2', \hat{f}_1\rangle \hat{f}_1\right)\right|\right| \leq M\phi \leq \frac{2K_1 M}{m} d(x_1,x_2).$$

Therefore, we have

$$||f(x_2) - f(x_1)|| \leq \left(\frac{2K_1 M}{m} + K_2\right) d(x_1,x_2),$$

so f is Lipschitz continuous. □

Remark. *We use this lemma to study the Lipschitz continuity of a vector flow field projected onto some regular boundary, which allows for the existence of a solution at the boundary that follows the projected flow field almost everywhere. We formalize this discussion in Section A.5.2.*

In Section A.8, techniques of linear algebra are used extensively to relate the dynamics of the voltage flow field to the trajectory of the auxiliary variable, so to conclude this Section, we provide the following useful lemma in anticipation.

Lemma A.4.4 (Continuity of Linear Maps)**.** *Given a metric space X, let $v : X \mapsto \mathbb{R}^m$ be a Lipschitz continuous map with bounded image, and let $M : X \mapsto \mathbb{R}^{n\times m}$ be another Lipschitz continuous map with bounded image. Then $M(\mathbf{x}) \cdot v(\mathbf{x})$ is Lipschitz continuous, where v is treated as a column vector, M is treated as an $n \times m$ matrix.*

Proof. From lemma A.4.2, we see that every component of v and every element of M must be Lipscthiz continuous and bounded. Then every component of $M \cdot v$ is Lipschitz continuous and bounded as well, as the addition and multiplication of bounded Lipschitz continuous functions are also bounded Lipschitz continuous. Therefore, using lemma A.4.2 again in reverse, we see that $M \cdot v$ must be Lipschitz continuous. □

A.5 Existence and Uniqueness of Caratheodory Solution

As discussed in the main text (see also Eq. (A.17) in Section A.6), the flow field we have chosen to govern the dynamics of our memcomputing machines are discontinuous. This is due to the presence of the min function and the explicit enforcement of the bounds on the dynamics. Therefore, the existence and uniqueness of a classical solution to the ODEs is not guaranteed. We then require the construction of a Caratheodory solution, and show that such construction is well-defined and unique. A Caratheodory solution is formally defined as follows:

Definition A.5.1 (Caratheodory Solution)**.** *Let $V : \mathbb{R}^n \mapsto \mathbb{R}^n$, then a solution to the ODE $\dot{x} = V(x)$ is a Catheodory solution if it satisfies*

$$x(t) = x(t_0) + \int_{t_0}^{t} f\big(x(s)\big)\, ds, \quad \forall t > t_0,$$

where $\int$ denotes the Lebesgue integral.

Remark. *An equivalent definition states that the Caratheodory solution follows the vector field everywhere along the solution trajectory except for a subset of measure zero [Cor08].*

We construct the Caratheodory solution in a way such that the analytic trajectory is closely mimicked by the dynamics governed by numerical simulations. In particular, the memory dynamics are governed by a discontinuous flow field, where occasionally the discretized trajectories will oscillate at certain hyper-planes of discontinuities until they "escape" the planes when the fields become sufficiently regular to allow so. The analytic construction of the Caratheodory solution is given such that the oscillatory dynamics at these hyperplanes are accounted for in a similar fashion. An extended discussion of how the analytic trajectory is simulated effectively by forward Euler is given in Section A.6.1.

A.5.1 Patching Vector Fields

Before we discuss the construction of Caratheodory solutions, we first formally define the class of discontinuous vector fields of interest referred to as the *patchy* vector fields. As the name suggests, the vector field is the result of patching together two different vector fields in a way such that a Caratheodory solution is admitted. For ease of analysis, we first assume some regularity condition on the boundary at which the fields are patched together.

Definition A.5.2 (Regular Domain)**.** *Let $\Omega \subset \mathbb{R}^n$ a domain in Euclidean space. The domain is said to be regular if it is bounded, with its boundary $\partial\Omega$ being C^∞ diffeomorphic to an $n-1$ sphere.*

Remark. *A regular domain is equipped with an orientable boundary, where the unit normal vector* $\mathbf{n}$ *can be defined at every point to be pointing towards the exterior of the domain. From here on, we shall use* $\text{int}(\Omega)$ *to denote the interior of the domain, which is simply itself if it is open in $\mathbb{R}^n$. And we use Ω^c to denote its complement in $\mathbb{R}^n$, and* $\text{ext}(\Omega) = \Omega^c/\partial\Omega$ *to denote the exterior.*

For any vector field with domain $\partial\Omega$, there is a unique "projection" of the field onto the boundary, such that the projection is in the tangent bundle generated by $\partial\Omega$.

Definition A.5.3. *For $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$, we denote the parallel and orthogonal components of* $\mathbf{v}$ *with respect to* $\mathbf{w}$ *as follows*

$$\mathbf{v}_{\mathbf{w},\parallel} = \langle \mathbf{v}, \hat{\mathbf{w}} \rangle \, \hat{\mathbf{w}} \qquad \mathbf{v}_{\mathbf{w},\perp} = \mathbf{v} - \mathbf{v}_{\mathbf{w},\parallel}$$

where $\hat{\mathbf{w}} = \frac{\mathbf{w}}{|\mathbf{w}|}$.

Definition A.5.4 (Decomposition at Boundary)**.** *Let $\Omega \subset \mathbb{R}^n$ be a regular domain, and let* $\mathbf{n}(\mathbf{x})$ *be the unit normal vector of Ω at $\mathbf{x} \in \partial\Omega$. Let $V : \mathbb{R}^n \mapsto \mathbb{R}^n$ be some vector field, then we denote the decomposition of the vector field at the boundary, $V_{\partial\Omega,\parallel} : \partial\Omega \mapsto \mathbb{R}^n$ and $V_{\partial\Omega,\perp} : \partial\Omega \mapsto \mathbb{R}^n$, as follows*

$$V_{\partial\Omega,\parallel}(\mathbf{x}) = V(\mathbf{x})_{\mathbf{n}(\mathbf{x}),\perp} \qquad V_{\partial\Omega,\perp}(\mathbf{x}) = V(\mathbf{x})_{\mathbf{n}(\mathbf{x}),\parallel}$$

$\forall \mathbf{x} \in \partial\Omega$.

Lemma A.5.1. *If $V : \partial\Omega \mapsto \mathbb{R}^n$ is bounded above and Lipscthiz continuous, then $V_{\partial\Omega,\|}$ and $V_{\partial\Omega,\perp}$ are Lipschitz continuous as well.*

Proof. Note that since V is bounded, $\exists M, ||V(\mathbf{x})|| \leq M, \forall \mathbf{x} \in \partial\Omega$. Furthermore, it is clear that $\mathbf{n}(\mathbf{x})$ is bounded from below as $||\mathbf{n}(\mathbf{x})|| = 1$ by definition of a unit vector. It can also be easily shown that $\mathbf{n}(\mathbf{x})$ is Lipschitz continuous due to the regularity of Ω. Therefore, by using lemma A.4.3, we see that $V(\mathbf{x})_{\partial\Omega,\perp}$ is Lipschitz continuous, which implies that $V(\mathbf{x})_{\partial\Omega,\|} = V(\mathbf{x}) - V(\mathbf{x})_{\partial\Omega,\perp}$ is also Lipschitz continuous. □

Since a projected vector field is Lipscthiz continuous, it admits a classical solution on the boundary (see Lemma A.4.1). However, at some point the trajectory has to escape the boundary once the field outside the boundary admits it. This escape condition depends on the direction of the field relative to the curvature of the boundary (see Proposition A.5.6). It is difficult to give a general definition of curvature for high dimensional hyper-surfaces. However, the definition of a *directional* curvature is relatively straightforward.

Definition A.5.5 (Directional Curvature)**.** *Let $\Omega \subset \mathbb{R}^n$ be a regular domain. Given a point in the boundary $\mathbf{x_0} \in \partial\Omega$ and a vector field $V : \mathbb{R}^n \mapsto \mathbb{R}^n$. Let $\gamma(t) \in \partial\Omega$ be a trajectory such that $\exists t_\varepsilon$,*

$$\gamma(0) = \mathbf{x_0}, \qquad \dot{\gamma}(t) = V_{\partial\Omega,||}(\gamma(t)), \quad \forall t \in [0, t_\varepsilon).$$

We then define the m-th order directional curvature at point $\mathbf{x_0}$ with respect to V as

$$\kappa_V^{(m)}(\mathbf{x_0}) = \left(\frac{d^m}{dt^m}\mathbf{n}(\gamma(t))\right) \cdot \hat{V}_{\partial\Omega,||}(\mathbf{x_0}),$$

for $m \geq 0$. For notational compactness, we define

$$m_0(\mathbf{x_0}) = \inf\{m \,|\, \kappa_V^{(m)}(\mathbf{x_0}) \neq 0\}, \qquad \kappa_V'(\mathbf{x_0}) = \kappa_V^{m_0(\mathbf{x_0})},$$

as the lowest order curvature that does not vanish.

Remark. *Visually, the sign of* κ *is an indicator of whether the boundary curves outward or inward at point* $\mathbf{x_0}$ *along the projected direction of* $\mathbf{v}$*, and this informs whether the solution should exit to the interior* Ω *or the exterior* $\Omega^c/\partial\Omega$ *(see Theorem A.5.8). It is clear that* $\kappa_{\mathbf{v}}^{(m)}(\mathbf{p})$ *is well defined and Lipschitz continuous to all orders due to the regularity of* Ω*.*

This definition of the curvature informs the patching operation of two vector fields at the boundary.

Definition A.5.6. *Let* $\Omega \subset \mathbb{R}^n$ *be a regular domain, and* $V : \mathbb{R}^n \mapsto \mathbb{R}^n$ *be some vector field. For* $\mathbf{x} \in \partial\Omega$*, we define the function* $\psi_{\partial\Omega,V} : \partial\Omega \mapsto \{0,1\}$ *as follows*

$$\psi_{\partial\Omega,V}(\mathbf{x}) = \begin{cases} 1 & \textit{if} \quad \kappa'_V(\mathbf{x}) \leq 0, \\ 0 & \textit{otherwise}. \end{cases}$$

Similarly, we define the function $\phi_{\partial\Omega,V} : \partial\Omega \mapsto \{0,1\}$ *as follows*

$$\phi_{\partial\Omega,V}(\mathbf{x}) = \begin{cases} 1 & \textit{if} \quad \kappa'_V(\mathbf{x}) \geq 0, \\ 0 & \textit{otherwise}. \end{cases}$$

Remark. *Note that the definition of* ψ *and* ϕ *is symmetric with respect to the exchange of the interior and exterior of the domain* Ω*.*

Definition A.5.7 (Patching)**.** *Let* $\Omega \subset \mathbb{R}^n$ *be a smooth open domain, and* $V, W : \mathbb{R}^n \mapsto \mathbb{R}^n$ *be two*

distinct vector fields. We define the patching of the two vector fields with respect to domain Ω *as*

$$\mathcal{P}_\Omega(V,W)(\mathbf{x})$$
$$= \begin{cases} V(\mathbf{x}) & \text{if} \quad \mathbf{x} \in \Omega, \\ W(\mathbf{x}) & \text{if} \quad \mathbf{x} \in \text{ext}(\Omega), \\ V(\mathbf{x})_{\partial\Omega,\parallel} + W(\mathbf{x})_{\partial\Omega,\parallel} + \psi_{\partial\Omega,V}(\mathbf{x})V(\mathbf{x})_{\partial\Omega,\perp} + \phi_{\partial\Omega,W}(\mathbf{x})W(\mathbf{x})_{\partial\Omega,\perp} & \text{if} \quad \mathbf{x} \in \partial\Omega. \end{cases}$$

Remark. *Note that the vector field* $\mathcal{P}_\Omega(V,W)$ *is piecewise Lipschitz continuous, with its discontinuity being at the boundary* $\partial\Omega$*. We can refer to* V *as the interior vector field and* W *as the exterior vector field. Visually, we can view the patched field at the boundary* $\partial\Omega$ *as some form of "projection" of the interior field* V *and exterior field* W*.*

A.5.2 Solution in the Boundary

It is clear that the patched field $\mathcal{P}_\Omega(V,W)(\mathbf{p})$ is Lipschitz continuous in Ω and $\text{ext}(\Omega)$ separately. This implies that a classical solution to the ODE $\dot{\mathbf{x}} = \mathcal{P}_\Omega(V,W)(\mathbf{x})$ with initial value $\mathbf{x}_0 \in \Omega$ exists up to the boundary $\partial\Omega$ (and similarly for $\mathbf{x}_0 \in \text{ext}(\Omega)$). Naturally, we also have to discuss the existence of a classical solution with $\mathbf{x}_0 \in \partial\Omega$. To do so, we first make a preliminary definition that specifies two important subsets of $\partial\Omega$, relative to which we attach the start- and end-points of the solution segments.

Definition A.5.8. *Given a regular domain* $\Omega \subset \mathbb{R}^n$ *and two vector fields* $V,W : \mathbb{R}^n \mapsto \mathbb{R}^n$*, we denote* $D_1 = \{\mathbf{x} \in \partial\Omega \,|\, \psi_{\partial\Omega,V}(\mathbf{p}) = 0\}$ *and* $D_2 = \{\mathbf{x} \in \partial\Omega \,|\, \phi_{\partial\Omega,W}(\mathbf{p}) = 0\}$*.*

Remark. *Visually,* D_1 *describes a region of the boundary where the interior field points outward, and* D_2 *describes a region of the boundary where the exterior field points inward. This gives rise to an irregular region* $D_1 \cap D_2$ *where the two fields "collide" at the boundary, which generates a Lipschitz continuous field that admits a classical solution in the boundary.*

Lemma A.5.2 (Continuity in Boundary)**.** *Given a regular domain $\Omega \subset \mathbb{R}^n$ and two bounded Lipschitz continuous vector fields $V, W : \mathbb{R}^n \mapsto \mathbb{R}^n$, $D = D_1 \cap D_2$ is open with respect to $\partial\Omega$. Furthermore, the vector field $\mathcal{P}_\Omega(V,W)$ is Lipschitz continuous in D.*

Proof. From the definitions of κ and ψ (see definitions A.5.5 and A.5.6), we can express $\partial\Omega/D_1$ as the following intersection of countably many sets

$$\begin{aligned}\partial\Omega/D_1 =& \{\mathbf{x} \in \partial\Omega \,|\, \kappa_V^{(0)}(\mathbf{x}) \leq 0\} \cap \\ & \bigcap_{m=1}^{\infty} \{\mathbf{x} \in \partial\Omega \,|\, \kappa_V^{(m-1)}(\mathbf{p}) = 0 \wedge \kappa_V^{(m)}(\mathbf{p}) \leq 0\}.\end{aligned}$$

We first assume that $\partial\Omega/D_1$ is non-empty, otherwise $D_1 = \partial\Omega = D$ is clearly open. Note from corollary A.5.1 that $V_{\partial\Omega,\perp}$ is Lipschitz continuous in $\partial\Omega$, so the $\kappa_V^{(0)}(\mathbf{x}) = \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x})\rangle$ is a continuous mapping from $\partial\Omega$ to $\mathbb{R}$. Furthermore, $\partial\Omega$ is compact, so its image must also be compact, with the infinum denoted as $-C = \inf_{\mathbf{x}\in\partial\Omega}\{\kappa_V^{(0)}(\mathbf{x})\} \leq 0$. This means that $\{\mathbf{x} \in \partial\Omega \,|\, \kappa_V^{(0)}(\mathbf{x}) \leq 0\}$ is the preimage of the closed set $[-C, 0]$ under a continuous mapping, so it also must be a closed set itself. A similar proof applies for the $m > 1$ cases. Therefore, $\partial\Omega/D_1$ is the intersection of countably many closed subsets of $\mathbb{R}$, so it must also be closed, which implies that D_1 is open. We can similarly show that D_2 is also open, so D being the intersection of two open sets is open as well.

To show that the field $\mathcal{P}_\Omega(V,W)$ is Lipschitz continuous in D, we first begin by noting that $\psi_{\partial\Omega,V}(\mathbf{x}) = \phi_{\partial\Omega,W}(\mathbf{x}) = 0$, $\forall \mathbf{x} \in D$, which follows directly from the definition of D and definition A.5.6. Then from definition A.5.7, we see that $\mathcal{P}_\Omega(V,W)(\mathbf{x}) = V(\mathbf{x})_{\partial\Omega,\|} + W(\mathbf{x})_{\partial\Omega,\|}$, $\forall \mathbf{x} \in D$. From corollary A.5.1, we see that $V_{\partial\Omega,\|}$ and $W_{\partial\Omega,\|}$ are Lipschitz continuous vector fields in D, then $\mathcal{P}(V,W)$ is also Lipschitz continuous. □

Corollary A.5.2.1 (Solution in Boundary)**.** *Given a regular domain $\Omega \subset \mathbb{R}^n$ and two bounded Lipschitz continuous vector fields $V, W : \mathbb{R}^n \mapsto \mathbb{R}^n$, let $U(\mathbf{x}) = V(\mathbf{x})_{\partial\Omega,\|} + W(\mathbf{x})_{\partial\Omega,\|}$, there is a unique classical solution $\mathbf{x}(t, \mathbf{x_0})$ to the ODE $\dot{\mathbf{x}} = U(\mathbf{x})$ for any $\mathbf{x_0} \in \partial\Omega$.*

Proof. We here provide a brief proof sketch. We begin by treating $\partial\Omega$ as a $n-1$ dimensional differentiable manifold (equipped with the pullback of the Euclidean metric by the natural embedding $\partial\Omega \mapsto \mathbb{R}^n$), then $U : \partial\Omega \mapsto T\partial\Omega$ is clearly Lipschitz continuous on the manifold. This implies that there is a unique classical solution to the ODE $\dot{\mathbf{x}} = U(\mathbf{x})$ on the manifold $\partial\Omega$ (under some suitable connection). □

Proposition A.5.3 (Containment in Boundary)**.** *Given a regular domain $\Omega \subset \mathbb{R}^n$ and two bounded Lipschitz continuous vector fields $V, W : \mathbb{R}^n \mapsto \mathbb{R}^n$, denote $\mathbf{x}(t, \mathbf{x_0})$ as the classical solution to the ODE $\dot{\mathbf{x}} = U(\mathbf{x})$ with $\mathbf{x_0} \in D$, where U is defined in corollary A.5.2.1. If we restrict the solution to $t \in [0, t_0)$, where $t_0 = \inf\{t \geq 0 \,|\, \mathbf{x}(t, \mathbf{x_0}) \in \partial\Omega\}$, with ∂D being the boundary of D with respect to $\partial\Omega$, then $\mathbf{x}(t, \mathbf{x_0})$ is a classical solution to the ODE $\dot{\mathbf{x}} = \mathcal{P}(V, W)(\mathbf{x})$.*

Proof. From lemma A.5.8, we see that $\mathcal{P}_\Omega(V, W)(\mathbf{x}) = U(\mathbf{x})$, $\forall \mathbf{x} \in D$. Since $\mathbf{x}(t) \in D$, $\forall t \in [0, t_0)$, we have $\dot{\mathbf{x}}(t) = U(\mathbf{x}(t)) = \mathcal{P}(V, W)(\mathbf{x}(t))$, $\forall t \in [0, t_0)$. Furthermore, $\mathbf{x_0} \notin \partial D$ as D is open, so $t_0 \neq 0$. □

To conclude, we have shown that the patched field admits a classical solution in $D = D_1 \cap D_2$ at least up to some positive time t_0.

A.5.3 Solution in the Domain

In the previous Section, we have shown how a solution segment can be constructed in the boundary $\partial\Omega$. In this subsection, we focus on the construction of a solution in the interior Ω and exterior $\text{ext}(\Omega)$ to the ODE $\dot{\mathbf{x}}(t) = \mathcal{P}(V, W)(\mathbf{x})$. WLOG, we can assume that the initial point is in the interior (see the remark of Definition A.5.6).

There are three possibilities for the evolution of the trajectory. First, the trajectory never leaves the interior Ω. Second, the trajectory escapes to the exterior $\text{ext}(\Omega)$, intersecting the boundary $\partial\Omega$ as required by the Jordan-Brouwer separation theorem [Mun14]. Finally, the trajectory

hits the boundary $\partial\Omega$ and "returns" back to the interior Ω.

Clearly, in the first case, the trajectory is simply the classical solution to the ODE, $\dot{\mathbf{x}} = V(\mathbf{x})$, and in the last two non-trivial cases, the trajectory reaches the boundary $\partial\Omega$ at some point. We first begin by noting that if the trajectory were to reach the boundary, it must enter $\partial\Omega$ through its subset $\overline{D_1}$.

Proposition A.5.4. *Given a regular domain $\Omega \subset \mathbb{R}^n$ and two bounded Lipschitz continuous vector fields $V, W : \mathbb{R}^n \mapsto \mathbb{R}^n$, let $\mathbf{x}(t, \mathbf{x_0})$ be the solution to the ODE $\dot{\mathbf{x}} = V(\mathbf{x})$ with initial value $\mathbf{x_0} \in \Omega$. If the solution intersects the boundary $\partial\Omega$ at time $t_0 = \inf\{t > 0 \,|\, \mathbf{x}(t) \in \partial\Omega\}$, then $\mathbf{x}(t_0) \in \overline{D_1}$.*

Proof. We provide here a sketch of the proof. Note that $\mathbf{x} \in \overline{D_1}$ implies the condition $\langle V(\mathbf{x}), \mathbf{n}(\mathbf{x})\rangle \geq 0$, required at the point of intersection. This condition can be shown by the fact that the trajectory $\mathbf{x}(t)$ intersects the boundary $\partial\Omega$ from the interior, and $\mathbf{x}(t)$ is continuously differentiable and the boundary $\partial\Omega$ is smooth. □

Remark. *Similarly, the solution $\mathbf{x}(t, \mathbf{x_0})$ to the ODE $\dot{\mathbf{x}} = W(\mathbf{x})$ with $\mathbf{x_0} \in \text{ext}(\Omega)$ must intersect the boundary $\partial\Omega$ in $\overline{D_2}$.*

At this point, we have shown how a trajectory initialized in the interior Ω reaches the boundary $\partial\Omega$. In order for the trajectory to be extended, we also have to consider how a solution exits the boundary. In order to guarantee that the trajectory does not violate the patched vector field in a non-zero measure set, we have to carefully specify the direction at which the trajectory exits the boundary to avoid "collision" with the field. We first formally define the notion of existence for a Caratheodory solution in a manner that suits our purpose.

Definition A.5.9. *Given a regular domain $\Omega \subset \mathbb{R}^n$ and a bounded Lipschitz continuous vector field $V : \mathbb{R}^n \mapsto \mathbb{R}^n$, the solution $\mathbf{x}(t, \mathbf{x_0})$ to the ODE $\dot{\mathbf{x}} = V(\mathbf{x})$ is said to exist in Ω up to t_0 if $\exists t_0 > 0$ such that $\mathbf{x}(t) \in \Omega$ for $\forall t \in [0, t_0)$.*

Lemma A.5.5. *Given a regular domain $\Omega \subset \mathbb{R}^n$ and a bounded Lipschitz continuous vector field $V : \mathbb{R}^n \mapsto \mathbb{R}^n$, and a solution to the ODE $\dot{\mathbf{x}} = V(\mathbf{x})$ initialized at $\mathbf{x}_0 \in \overline{\Omega}$. Then the following statements are true:*

- *If $\mathbf{x}_0 \in \Omega$, then a solution always exists in Ω.*
- *If $\mathbf{x}_0 \in \partial\Omega$, then a solution exists in Ω if $\kappa'_V(\mathbf{x}_0) \leq 0$, and a solution does not exist if $\kappa'_V(\mathbf{x}_0) > 0$.*

Proof. The proof of the first statement is simple. We first let $\mathbf{x}(t)$ be a classical solution to the ODE $\dot{\mathbf{x}} = V(\mathbf{x})$ initialized at $\mathbf{x}_0 \in \Omega$. Note that since Ω is open, it is possible to find an open ball in Ω, $B_\delta(\mathbf{x}_0) \subset \Omega$, centered at $\mathbf{x}_0$ with radius δ. Since $\mathbf{x}(t)$ is continuous with respect to t, it is possible to find a $t_\varepsilon > 0$ such that $\mathbf{x}(t) \in B_\delta(\mathbf{x}_0)$ for $\forall t \in (0, t_\varepsilon)$. Therefore, we see that $\mathbf{x}(t)$ exists in Ω.

The proof of the second statement is more involved, and we here only provide a proof sketch. We first let $\mathbf{x}(t)$ be a classical solution to $\dot{\mathbf{x}} = V(\mathbf{x})$ initialized at $\mathbf{x}_0 \in \partial\Omega$. We can then express a small neighborhood of $\mathbf{x}_0$ as a graph of some analytic function $f : \mathbb{R}^{n-1} \mapsto \mathbb{R}$. We can then "project" the trajectory $\mathbf{x}(t)$ onto the boundary, and denote its projection as $\mathbf{x}'(t)$. We can time-evolve the trajectory and its projection simultaneously forward infinitesimally by δt. We can find the displacement between the solution trajectory and its projection along the direction of the normal vector, $\langle \mathbf{x}(\delta t) - \mathbf{x}'(\delta t), \mathbf{n}(\mathbf{x}_0) \rangle$, and expand it in terms of δt into a convergent series. If the series converge into a negative number, then the trajectory is able to "enter" the domain Ω, so a solution exists in the domain. On the other hand, if the series converge into a positive number, then the trajectory can only "leave" the domain Ω, so a solution does not exist. □

Remark. *The visual interpretation of this lemma is rather straightforward. It essentially states that a trajectory initialized at the boundary of a regular domain can enter into the interior only if the field points inward at that point.*

If the trajectory is initialized in D, then the trajectory clearly must remain in the boundary as discussed in the remark of Lemma A.5.8. Therefore, the trajectory can exit the boundary only if $\mathbf{x}_0 \notin D$, or $\big(\kappa'_V(\mathbf{x}_0) \leq 0\big) \vee \big(\kappa'_W(\mathbf{x}_0) \geq 0\big)$, in which case a solution exists in the interior and exterior respectively.

Proposition A.5.6 (Exiting the Boundary)**.** *Given a regular domain $\Omega \subset \mathbb{R}^n$ and two bounded Lipschitz continuous vector fields $V, W : \mathbb{R}^n \mapsto \mathbb{R}^n$, we let the initial condition be $\mathbf{x}_0 \in \partial\Omega/D$, then a solution to $\dot{\mathbf{x}} = \mathcal{P}(V,W)(\mathbf{x})$ can be uniquely constructed as:*

- *The classical solution to $\dot{\mathbf{x}} = V(\mathbf{x})$ at least up to some positive time if $\kappa'_V(\mathbf{x}_0) \leq 0$.*
- *The classical solution to $\dot{\mathbf{x}} = W(\mathbf{x})$ at least up to some positive time if $\kappa'_W(\mathbf{x}_0) \geq 0$ and $\kappa'_V(\mathbf{x_0}) > 0$.*

Proof. The proof follows directly from Definition A.5.7 and Lemma A.5.5. □

Remark. *To interpret this proposition visually, we imagine a point in the boundary such that either the interior field or the exterior field points away from the boundary. If the interior field points away from the boundary, then the trajectory should enter Ω from $\partial\Omega$, and the trajectory will "follow" the field initially, as both the trajectory and the interior field point inward with respect to the domain Ω. Similarly, if the exterior field points away, then the trajectory should enter $\text{ext}(\Omega)$ instead. If both fields point away from the boundary, then the trajectory has a choice of entering either Ω or $\text{ext}(\Omega)$, and we let the trajectory enter Ω as the convention.*

A.5.4 Bridging the solutions

Up to this point, we have shown how a Caratheodory solution can be constructed in an open domain Ω and its boundary $\partial\Omega$, and we are now ready to construct the *maximal* Caratheodory solution that is capable of traversing all three domains: Ω, and $\partial\Omega$, and $\text{ext}(\Omega)$. WLOG, we can assume that the initial value $\mathbf{x}_0 \in \Omega$ to be in the interior, then there are three possibilities for the

time evolution of the trajectory. Essentially, the maximal Caratheodory solution is constructed as the extension of a classical solution in one domain with another classical solution in another domain. A formal description of the maximal solution is presented as a constructive proof of Theorem A.5.8 based on the formal definition of *extension* as follows.

Definition A.5.10. *Given a set X and two functions, $\mathbf{x}_1 : [0,t_1] \mapsto X$, $\mathbf{x}_2 : [0,t_2] \mapsto X$, we say that $\mathbf{x}_2(t)$ is an extension of $\mathbf{x}_1(t)$ if $t_2 > t_1$, and $\mathbf{x}_2(t) = \mathbf{x}_1(t)$ for $\forall t \in [0,t_1]$. Alternatively, we can say that $\mathbf{x}_1(t)$ is extended with $\mathbf{x}_2(t-t_1)$ at point $\mathbf{x}_1(t_1)$.*

Lemma A.5.7. *Let $\mathbf{x}_1 : [0,t_1] \mapsto \mathbb{R}^n$ and $\mathbf{x}_2 : [0,t_2] \mapsto \mathbb{R}^n$ be two Caratheodory solutions to the ODE $\dot{\mathbf{x}} = F(\mathbf{x})$ where $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ is some vector field. If $\mathbf{x}_2(0) = \mathbf{x}_1(t_1)$, then we can extend $\mathbf{x}_1(t)$ with $\mathbf{x}_2(t)$, which results in another Caratheodory solution to the ODE.*

Proof. This is obvious if we note that the procedure of attaching the two solution segments will result in potentially violating the ODE only at a single point $\mathbf{x_1}(t_1)$. □

Theorem A.5.8 (Construction of Maximal Caratheodory Solution)**.** *Given an open regular domain $\Omega \in \mathbb{R}^n$ and two bounded Lipschitz continuous vector fields $V, W : \mathbb{R}^n \mapsto \mathbb{R}^n$, it is possible to construct a unique Caratheodory solution to the ODE $\dot{\mathbf{x}} = \mathcal{P}(V,W)(\mathbf{x})$, where $\mathcal{P}$ is the patching operation defined in Definition A.5.7.*

Proof. WLOG, we assume that the initial value of the ODE is $\mathbf{x}_0 \in \Omega$. Let $\mathbf{x}_1(t)$ be the classical solution to $\dot{\mathbf{x}} = V(\mathbf{x})$ existing up to t_1 in Ω. If $t_1 = +\infty$, then $\mathbf{x}_1(t)$ is trivially a Caratheodory solution as well. We then consider the case where t_1 is finite, meaning that the trajectory enters the boundary $\partial\Omega$ at some point $\mathbf{p} = \mathbf{x}_1(t_1) \in \overline{D_1}$ (see Proposition A.5.4). Note that $\kappa'_V(\mathbf{p}) \geq 0$, so we are left with the following cases:

- If $\kappa'_W(\mathbf{p}) \geq 0$, then we extend $\mathbf{x}_1(t)$ with the maximal classical solution to the ODE $\dot{\mathbf{x}} = W(\mathbf{x})$ in $\text{ext}(\Omega)$ initialized at $\mathbf{p}$. The extended solution violates the ODE only at $\mathbf{p}$.

- If $\kappa'_W(\mathbf{p}) < 0$ and $\kappa'_V(\mathbf{p}) = 0$, then we extend $\mathbf{x}_1(t)$ with the maximal classical solution to the ODE $\dot{\mathbf{x}} = V(\mathbf{x})$ in Ω initialized at $\mathbf{p}$. The extended solution violates the ODE only at $\mathbf{p}$.

- If $\kappa'_W(\mathbf{p}) < 0$ and $\kappa'_V(\mathbf{p}) > 0$, then $\mathbf{p} \in D$, and let $\mathbf{x}_2(t)$ be the maximal classical solution to the ODE $\dot{\mathbf{x}} = U(\mathbf{x})$ existing in D up to t_2, where $U(\mathbf{x})$ is defined in corollary A.5.2.1. If $t_2 = +\infty$, then we are done; if t_2 is finite, then we let $\mathbf{q} = \mathbf{x}_2(t_2) \in \partial D$, implying that $\kappa'_V(\mathbf{q}) = 0$ or $\kappa'_W(\mathbf{q}) = 0$, reducing to the previous two cases. The extended solution violates the ODE only at $\mathbf{p}$ and $\mathbf{q}$.

We iterate this procedure every time the trajectory enters the boundary, with the treatment of the entrance from the exterior $\text{ext}(\Omega)$ mirroring the entrance from interior Ω. This gives us the maximal Caratheodory solution if we take $t \to \infty$. It is clear that the solution can only be extended countably many times, and each segment is classical in nature (see Proposition A.5.6) meaning that the ODE is only violated at countably many points, so the maximal solution is in fact Caratheodory by Definition A.5.1.

□

Remark. *Visually, for a trajectory initialized in Ω that enters the boundary $\partial\Omega$, we have three scenarios. In the first scenario, the trajectory is guided by the interior field in a way such that it barely "scrapes" the boundary and returns back to the interior. In the second scenario, the trajectory "crosses" the boundary and continues its path into the exterior if the exterior field at the intersection points outward. Finally, if the trajectory enters into the boundary at a point where the interior and exterior fields both point inward, then the trajectory "tunnels" in the boundary to avoid the two fields and continues to do so until it reaches a point where one of the two fields begins pointing outward, then the trajectory begins to follow that field. If both fields never point outward, then the trajectory remains in the boundary forever.*

This concludes the section which establishes the necessary mathematical formalism for

discussing the memcomputing dynamics which are guided autonomously by such patchy vector fields (see Eq. (A.17)).

A.6 Memory Dynamics

To find an assignment $\mathbf{v}$ that minimizes the constraint in Eq. (A.8), we can time-evolve the voltages autonomously with the following ODE (which is an equivalent way of writing Eqs. (A.1)-(A.6)):

$$\begin{aligned}
\dot{v}_i &= \sum_{j=1}^{m} \Big\{ \frac{1}{2} x_{l,j} x_{s,j} q_{ij} \min_{\{i' \neq i \,|\, q_{i'j} \neq 0\}} (1 - q_{i'j} v_{i'}) + (1 + \zeta x_{l,j})(1 - x_{s,j}) \delta_{i\sigma_j} q_{ij} C_j(\mathbf{v}) \Big\}, \\
\dot{x}_{s,j} &= \beta\big(x_{s,j} + \varepsilon\big)\big(C_j(\mathbf{v}) - \gamma\big), \\
\dot{x}_{l,j} &= \alpha\big(C_j(\mathbf{v}) - \delta\big).
\end{aligned} \tag{A.17}$$

From now on, we shall refer to this particular ODE as *memory dynamics*, where $\mathbf{v} \in [-1,+1]^n$ are voltages corresponding to the Boolean variables of the original 3-SAT problem with n variables and m clauses. Furthermore, we refer to $\mathbf{x_s} \in [0,1]^m$ as *short-term memory* and $\mathbf{x_l} \in [1, x_{max}]^m$ as *long-term memory*, where $x_{max} > 1$ is some upper bound to the slow variable dynamics[8]. The parameters $\{\alpha, \beta, \gamma, \delta, \zeta, \varepsilon\}$ are positive constants empirically tuned to provide the regularity and convergence of the dynamics with a sufficiently fast time scale (see Sec. A.2.1). We will use the non-subscript symbol, $\mathbf{x} = \{\mathbf{v}, \mathbf{x_s}, \mathbf{x_l}\} \in \mathbb{R}^{n+2m}$, to denote the collection of all dynamic variables, allowing us to write the ODE as

$$\dot{\mathbf{x}} = F(\mathbf{x}),$$

where F is some flow field corresponding to the RHS of Eqs. (A.17).

[8] Note that the bounds on the dynamic variables $\{\mathbf{v}, \mathbf{x_f}, \mathbf{x_s}\}$ are not enforced "naturally" by the memory dynamics. They are enforced through the introduction of auxiliary fields in the exterior of the bounded domain. See Section A.6.2 for a formal discussion of the procedure of doing so.

For the sake of having a more compact expression for the ODE equations, it is convenient for us to borrow the notation of Eq. (A.5) and denote

$$G_{ij}(\mathbf{v}) = \frac{1}{2} q_{ij} \min_{\{i' \neq i \,|\, q_{i'j} \neq 0\}} (1 - q_{i'j} v_{i'}) \tag{A.18}$$

as the *gradient-like* term, as it approximately follows the directional gradient of the energy of the j-th clause along the direction of v_i (see Eq. (A.10)). Note that the actual directional gradient is similar to Eq. (A.18) with the only exception being the min operation replaced with the product $\prod$. The magnitude of the gradient-like term for a voltage in the j-th constraint is related to the value of the other two voltages in the constraint. Similarly, we borrow the notation of Eq. (A.6) and denote

$$R_{ij}(\mathbf{v}) = \delta_{i\sigma_j} q_{ij} C_j(\mathbf{v}) \tag{A.19}$$

as the *rigidity* term. Its magnitude is equivalent to the clause constraint C_j defined in Eq. (A.7) if v_i is the voltage that defines C_j, and zero otherwise.

We can then succinctly write the voltage dynamics as

$$\dot{\mathbf{v}} = \mathbf{G}(\mathbf{v})(\mathbf{x_s} * \mathbf{x_l}) + \mathbf{R}(\mathbf{v})\big((1 + \zeta \mathbf{x_l}) * (1 - \mathbf{x_s})\big), \tag{A.20}$$

where $\mathbf{G}$ and $\mathbf{R}$ are treated as $n \times m$ matrices dependent on $\mathbf{v}$, the operator $*$ denotes element-wise multiplication, and $\mathbf{x_s}$ and $\mathbf{x_l}$ are treated as column vectors for the sake of matrix operation. In this form, we can clearly see that the gradient-like and rigidity dynamics are weighted clause-wise by the memory variables. The presence of dynamic memory is a central feature of our dynamics.

For certain analyses of dynamical properties, it is sufficient and more convenient for us to

focus on the analytic properties of the following simplified dynamics

$$\begin{aligned} \dot{\mathbf{v}} &= \mathbf{G}(\mathbf{v})\mathbf{x_l}, \\ \dot{\mathbf{x}}_\mathbf{l} &= \alpha \mathbf{C}(\mathbf{v}), \end{aligned} \tag{A.21}$$

All the dynamical properties derived in this work under the assumption of this simplified dynamics can be easily generalized to the full dynamics if we assume sufficiently general forms for **G** and **C** (see Section A.8 and A.9).

A.6.1 Discontinuous Hyperplanes

We first make the important observation that the gradient-like term **G** is not differentiable everywhere and the rigidity term **R** is not Lipschitz continuous. These irregular points form hyperplanes generated by the minimum operation in the voltage space. In this section, we construct the hyperplanes which contain all the points of discontinuity for the rigidity term. These hyperplanes are generated by the binary values of $\delta_{i\sigma_j}$ (which contains implicitly a minimum operation), and they form $n-1$ dimensional hyperplanes in the voltage space $\mathbb{R}^n$. A similar construction also applies for the gradient-like term[9].

Proposition A.6.1 (Hyperplanes)**.** *There exists a union of countably many $(n-1)$-dimensional hyperplanes in $\mathbb{R}^n$ such that it contains all the points where the field F is discontinuous.*

Proof. To lessen the burden of notation, we let $N = [[1,n]]$ and $M = [[1,m]]$. We first recall from Eq. (A.9) that

$$\sigma_j = \text{argmin}_{\{i \,|\, q_{ij} \neq 0\}} (1 - q_{ij} v_i),$$

which implies that the field can only be discontinuous at a point where some j can be chosen such

[9]Finding these hyperplanes for the gradient-like term is not strictly necessary, as the gradient-like term is already Lipschitz continuous. The hyperplanes will only contain points of non-differentiability, which will not affect the existence and uniqueness of the dynamical trajectory (see Section A.5).

that the argmin operation is degenerate, which is equivalent to the following condition

$$\exists j \in M, \exists i_1, i_2 \in \{i \in N \,|\, q_{ij} \neq 0\}, q_{i_1 j} v_{i_1} = q_{i_2 j} v_{i_2}. \tag{A.22}$$

We denote the set of all points $\mathbf{x}$ that satisfies the above condition as $\partial\Omega$.

For any two distinct indices of the Boolean variables, or $\forall i_1, i_2 \in N$ where $i_1 \neq i_2$, we can define a *positive* hyperplane H_P and a *negative* hyperplane H_N as follows

$$H_{P,i_1 i_2} = \{\mathbf{x} \in \mathbb{R}^{n+2m} \,|\, v_{i_1} = v_{i_2}\},$$
$$H_{N,i_1 i_2} = \{\mathbf{x} \in \mathbb{R}^{n+2m} \,|\, v_{i_1} = -v_{i_2}\}.$$

Note that both are $(n-1)$-dimensional. If we recall that $q_{ij} = \pm 1$ for all nonzero elements of the polarity matrix Q, then it can be shown that any voltage assignment $\mathbf{v}$ that satisfies condition (A.22) must be in one of such hyperplanes. Therefore, the union of all such hyperplanes must contain $\partial\Omega$, or

$$\partial\Omega \subseteq \bigcup_{i_1 \neq i_2} \left(H_{P,i_1 i_2} \cup H_{N,i_1 i_2}\right).$$

Note that there are $\binom{n}{2}$ positive and negative hyperplanes each, so there are $2\binom{n}{2}$ hyperplanes in total, which is a countable number. This proves the proposition. □

Remark. *An immediate consequence of this proposition is that the rigidity term is only discontinuous at a measure zero subset of the phase space, as all the hyperplanes of discontinuities are of measure zero, and there are only countably many of them. Therefore, the rigidity term is smooth almost everywhere. Note that these hyperplanes also contain the points at which the gradient-like term is non-differentiable, meaning that the gradient-like term is also smooth almost everywhere.*

As the field is continuous almost everywhere, it clearly admits a Caratheodory solution for any initial value, if the fields are patched appropriately at the hyperplanes according to the procedure in Definition A.5.7. Note that the phase space of the dynamics is an $n+2m$-dimensional hypercube (see Section A.6.2), which is partitioned into disjoint subsets by the hyperplanes. A caveat here is that the domains are *almost* regular as the intersections of the hyperplanes generate regions of non-smoothness. However, note that these intersections have zero measure relative to the hyperplanes, so it is unlikely for a trajectory to encounter them. For the sake of analytic completeness, even if we assume that a trajectory were to encounter an intersection of planes, this does not invalidate our method of constructing a Caratheodory solution, as there is still a unique projection of vector fields on these intersecting regions. As for using the directional curvature as the exit condition, the zeroth order directional curvature can be defined as $\pm\infty$ accordingly at these regions, and the exit protocol as given in proposition A.5.6 remains unchanged.

A.6.2 Compact Positive Invariant Set

To respect the Boolean structure of the original 3-SAT problem, the dynamics as given in Eqs. (A.17) must be bounded explicitly. First of all, we choose the bound the voltages explicitly in a compact set, which is $[-1,+1]^n$ for our work[10]. Furthermore, the short-term memory $\mathbf{x_s}$ has to be bounded in $[0,1]^m$, as a way to completely stop either the gradient-like or rigidity contribution to the dynamics for each clause. Finally, the long-term memory $\mathbf{x_l}$ has to be bounded in $[1,x_{\max}]^m$ in practice[11]. In fact, for the analysis in the following sections, we will regularly assume that the bound $x_{\max}$ on the long-term memory is absent, meaning that $\mathbf{x_l} \in [1,+\infty)^m$, in an effort to increase the generality of certain propositions. The bounds on the short-term memory is crucial,

[10]Note the choice of -1 and $+1$ is to make an intuitive connection to the false and true state. As $\dot{\mathbf{v}}$ and $\mathbf{C}(\mathbf{v})$ are linear with respect to $\mathbf{v}$, the lower and upper bound for the voltages can be chosen arbitrarily (centered at 0), and the original dynamics can be recovered via an appropriate rescaling of the memory variables and constant parameters.

[11]This is mostly for the sake of a practical implementation of our solver. Note that if the upper bound is absent, then a digital implementation would require infinite precision and an analogue implementation would require infinite energy to guarantee accurate simulation, neither of which is possible.

however, and will always be assumed present.

Putting everything together, this means that the dynamics must be fully contained within the region $O = [-1,1]^n \times [0,1]^m \times [1, x_{\max}]^m$, which is a compact set in $\mathbb{R}^{n+2m}$. To put this formally, we have to show that O is an invariant set, and any trajectory with initial value in O must remain in O forever. To do so, we consider a general ODE with the flow field defined in a regular domain Ω, such that a Caratheodory solution exists in the domain. In other words, we have $\dot{\mathbf{x}} = F(\mathbf{x})$, where $F : \mathbb{R}^{n+2m} \mapsto \mathbb{R}^{n+2m}$ is some sufficiently regular vector field in $\Omega \subset \mathbb{R}^{n+2m}$. Suppose we now wish to modify the vector field in such a way that, for any initial value $\mathbf{x}(0) \in \Omega$, the trajectory is contained entirely within the closure of that domain $\overline{\Omega}$, or $\mathbf{x}(t) \in \overline{\Omega}$ for $\forall t \geq 0$. This has to be done carefully such that the original flow field in Ω remains the same. We do so by patching the original vector field with a "bounding" vector field in $\text{ext}(\Omega)$ as follows.

Lemma A.6.2 (Bounding Field)**.** *Let $\Omega \subset \mathbb{R}^n$ be a smooth open domain, and let $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ be some bounded vector field that admits a Caratheodory solution in Ω. Let $G : \mathbb{R}^n \mapsto \mathbb{R}^n$ be some Lipschitz continuous vector field satisfying*

$$\forall \mathbf{x} \in \partial\Omega, \quad G(\mathbf{x}) = -M\mathbf{n}(\mathbf{x}),$$

where $M > 0$ can be any positive constant, and $\mathbf{n}(\mathbf{x})$ is the outward pointing unit normal vector of the boundary $\partial\Omega$ at $\mathbf{x}$. Then any construction of the Caratheodory solution (see theorem A.5.8) to the ODE, $\dot{\mathbf{x}} = \mathcal{P}(F,G)(\mathbf{x})$, with initial value $\mathbf{x_0} \in \Omega$, has the property that $\mathbf{x}(t) \in \overline{\Omega}$ for $\forall t \geq 0$.

Proof. Note that based on the construction given in theorem A.5.8, it is sufficient to show that $\kappa'_G(\mathbf{x}) < 0$ for $\forall \mathbf{x} \in \partial\Omega$, as the trajectory will never be able to exit into the region $\text{ext}(\Omega)$. By construction, we have $\langle G(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle = -M$ for $\forall \mathbf{x} \in \partial\Omega$, so it follows directly from definition A.5.5 that $\kappa'_G(\mathbf{x}) = -M < 0$. □

Remark. *By adding the "bounding" vector field G, we are essentially "projecting" any "stray"*

fields onto the boundary $\partial\Omega$, such that whenever a trajectory enters the boundary, it will continue to "flow" inside the boundary (see corollary A.5.2.1) and never escape $\overline{\Omega}$. An important point to note is that the dynamics do not stop after reaching $\partial\Omega$.

Corollary A.6.2.1 (Invariant Hypercube)**.** *Let $O = [-1,1]^n \times [0,1]^m \times [1, x_{\max}]^m \subset \mathbb{R}^{n+2m}$, and let $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ be some bounded vector field that admits a Caratheodory solution in O. For $\forall i \in [[1, n+2m]]$, we let k_i be the lower bound of the i-th quotient space of O, and let K_i be the upper bound. Then we define the left and right hyperplanes, L_i and R_i, as follows*

$$L_i = \{\mathbf{x} \in O \,|\, x_i = k_i\} \qquad R_i = \{\mathbf{x} \in O \,|\, x_i = K_i\}.$$

Let $G : \mathbb{R}^n \mapsto \mathbb{R}^n$ be some Lipschitz continuous vector field such that $\forall i \in [[1, n+2m]]$:

$$\forall \mathbf{x} \in L_i, \quad G(\mathbf{x}) = M\hat{\mathbf{e}}_i$$
$$\forall \mathbf{x} \in R_i, \quad G(\mathbf{x}) = -M\hat{\mathbf{e}}_i,$$

where $M > 0$ can be any positive constant, and $\hat{\mathbf{e}}_i$ is the i-th component of the standard basis. Then O is a positive invariant set under the ODE, $\dot{\mathbf{x}} = \mathcal{P}(F,G)(\mathbf{x})$. Furthermore, the superposed flow field on the hyperplanes is given by

$$\forall \mathbf{x} \in L_i, \quad \mathcal{P}(F,G)(\mathbf{x}) = F(\mathbf{x}) - F_i(\mathbf{x})\big(1 - H(x_i)\big)\hat{\mathbf{e}}_i,$$
$$\forall \mathbf{x} \in R_i, \quad \mathcal{P}(F,G)(\mathbf{x}) = F(\mathbf{x}) - F_i(\mathbf{x})H(x_i)\hat{\mathbf{e}}_i,$$

where H denotes the Heaviside step function.

Remark. *To visualize the bounding flow field, one can imagine a hypercube O where the internal field remains unchanged, and the exterior field is "pressing against" the faces of the cube to ensure that any trajectory initialized inside the cube remains inside. The flow field on the "faces" of the cube is simply the projection of the field onto the plane if the field were to point outward.*

This bounding procedure effectively mimics the numerical technique that we use to bound the dynamics, where any outward pointing component of the flow field on the boundary is simply ignored.

From now on, when we refer to *memory dynamics*, we are referring to the system of ODEs given in Eqs. (A.17), with the bounds of the dynamics enforced by the exterior field G as constructed in corollary A.6.2.1. To lessen the burden of notation, we shall refer to the patched flow field of the memory dynamics, $\mathcal{P}(F,G)$, simply as F. The set $O = [-1,1]^n \times [0,1]^m \times [1,x_{\max}]^m$ is then a positive invariant set of the memory dynamics.

A.6.3 Gauge Invariance of Dynamics

In this Section, we primarily focus on formalizing the notion of gauge invariance for the dynamics governed by Eqs. (A.17). To do so, it is convenient to first reformulate the flow field as a group action.

Definition A.6.1 (Time Mapping)**.** *Given a vector field $V : \mathbb{R}^n \mapsto \mathbb{R}^n$ such that there is a unique positive solution $\mathbf{x}(\mathbf{x_0},t)$ to the ODE $\dot{\mathbf{x}} = F(\mathbf{x})$ for any initial value $\mathbf{x_0} \in \mathbb{R}^n$, we define a mapping $T_s : \mathbb{R}^n \mapsto \mathbb{R}^n$ for $\forall s \geq 0$ as follows*

$$T_s(\mathbf{x}_0) = \mathbf{x}(s, \mathbf{x}_0).$$

Remark. *It should first be noted that T_s is a well defined operator $\forall s \geq 0$, as the solution to the ODE with any initial value is unique. It can also be easily checked that the operators T_s form a semigroup with the identity element being T_0. In fact, we have*

$$T_{s_2} T_{s_1}(\mathbf{x}_0) = T_{s_2}\big(\mathbf{x}(s_1,\mathbf{x}_0)\big) = \mathbf{x}\Big(s_2, \big(\mathbf{x}(s_1,\mathbf{x}_0)\big)\Big) = \mathbf{x}(s_1+s_2,\mathbf{x}_0) = T_{s_1+s_2}(\mathbf{x}_0).$$

The reason why the operators form only a semigroup is because it does not necessarily have a

group inverse, as we do not require the negative solution to the ODE to exist or be unique.

For our memory dynamics, an important property of T_s is that it is invariant under gauge conjugation. This is important as it essentially allows us to simplify the analysis of the memory dynamics by assuming that a solution vector is $\mathbf{v}_0 = +\mathbf{1}$.

Proposition A.6.3 (Gauge Invariance of Dynamics)**.** *Given a polarity matrix Q corresponding to a satisfiable 3-SAT instance with some solution vector $\mathbf{v}_0$, and an operator T_s corresponding to the memory flow field F, we have the following*

$$T_s = G_{\mathbf{v}_0} \circ T_s \circ G_{\mathbf{v}_0}^{-1},$$

where $G_{\mathbf{v}_0}$ is the gauge mapping operation in Definition A.3.3.

Remark. *We here provide a proof sketch of this proposition. We first begin by noting that the operators T_s form a semigroup, so it is sufficient to show that the infinitesimal group generator F is invariant under gauge conjugation, or*

$$F = G_{\mathbf{v}_0} \circ F \circ G_{\mathbf{v}_0}^{-1}.$$

This is equivalent to showing that transforming both the LHS and RHS of the equations in (A.17) does not violate the equalities, which can be easily shown by recalling that $C_j(\mathbf{v})$ is gauge invariant (see lemma A.9.4.1). Then we see that a prefactor of $v_{0,i}$ appears in both the LHS and RHS of the voltage equations. Furthermore, we can also easily show that the discontinuous hyperplanes (see Section A.6.1) and the boundaries of the hypercube containing the dynamics (see corollary A.6.2.1) are also invariant under the gauge mapping. Therefore, the operator T_s must be invariant under gauge conjugation for $\forall s \geq 0$.

A.6.4 Correspondence between Fixed Points and Solutions

The fixed points of the dynamics must correspond to a solution to the original 3-SAT instance (if the instance is satisfiable). Otherwise, the correctness of the memory dynamics as a SAT solver cannot be guaranteed, as it is possible for the dynamics to terminate at a point corresponding to a non-solution. We dedicate this section to the correspondence between the fixed points of the dynamics and the solutions of a 3-SAT instance. Before we continue this discussion, we first note that it is possible to solve a 3-SAT Boolean formula with a partial assignment of the Boolean variables, which corresponds to hyperfaces on the voltage hypercube (see Section A.3.3). In other words, it is possible for the dynamics to solve a 3-SAT instance by converging to a hyperface instead of any particular solution vector, and the solution can be extracted by choosing an arbitrary vertex of that hyperface.

Definition A.6.2 (Solution Plane)**.** *Consider a 3-SAT problem defined by an $n \times m$ polarity matrix. If we can find a non-empty subset of indices, $I \in [[1,n]]$, such that there are exactly $2^{|I|}$ distinct solutions coinciding to the assignment of the Boolean variables indexed $[[1,n]]/I$, then the problem is said to be partially solvable, and we refer to $I = [[1,n]]/I$ as the isolated index set of the solutions. I is said to be proper if it has no proper subset that is also an isolated index set.*

Let $\mathbf{v}'$ be a solution vector, and I be a proper index set. We define the solution plane to be

$$H(\mathbf{v}', I) = \{\mathbf{v} \in [-1,+1]^n \;\big|\; \forall i \in I,\, v_i = v_i'\}.$$

The vertices (which are solution vectors) are said to be connected by this plane. Any solution vector that is not connected by a solution plane is said to be isolated.

Remark. *Note that for a given solution vector $\mathbf{v}'$, its proper index set is not necessarily unique, and depends on the polarity matrix of the 3-SAT Boolean formula. The solution plane is, however, unique given a solution vector and its proper index set.*

Lemma A.6.4. *Let* $\mathbf{v}'$ *be a solution vector for which a proper index set* I *exists. Then*

$$\forall \mathbf{v} \in H(\mathbf{v}', I), \qquad \mathbf{C}(\mathbf{v}) = \mathbf{0}.$$

On the other hand, let $\mathbf{v}$ *be a vector such that* $\mathbf{C}(\mathbf{v}) = 0$*, and* $\mathbf{v}' = \mathrm{sgn}(\mathbf{v})$ *be the corresponding solution vector. If a proper index set* I *exists for the solution vector, then*

$$\exists I, \qquad \mathbf{v} \in H(\mathbf{v}', I).$$

Proof. The proof follows trivially from the definition of the solution plane (see definition A.6.2) and the definition of the clause constraint (see Eq. (A.7)). □

Remark. *One immediate implication of this lemma is that once we have found a voltage assignment such that the global constraint (or energy) is zero, then the voltage vector must be either a solution vector, or it must be in some solution plane. If it is in a solution plane, then we can take any vertex of that plane as a solution to the 3-SAT problem.*

Since a solution vector and a vector in a solution plane both solve the 3-SAT problem, we can treat a solution vector equivalently to a solution plane. Then for an isolated solution vector $\mathbf{v}'$, its solution plane simply refers to itself.

Proposition A.6.5 (Solution Fixed Points)**.** *If* $\mathbf{v}'$ *is in a solution plane, then* $\mathbf{x}' = \{\mathbf{v}', \mathbf{x_s}, \mathbf{x_l}\}$ *will eventually evolve to a fixed point in the same solution plane* $\forall \mathbf{x_s} \in [0,1]^m$*,* $\forall \mathbf{x_l} \in [1, +\infty)^m$*. Conversely, if* $\mathbf{x}'$ *is a fixed point of the memory dynamics, then* $\mathbf{v}' = \{x'_1, ..., x'_n\}$ *is in a solution plane.*

Proof. We first show the first part of the proposition. Given any $\mathbf{x_s}$ and $\mathbf{x_l}$, we denote $\mathbf{x}' = \{\mathbf{v}', \mathbf{x_f}, \mathbf{x_s}\}$, where $\mathbf{v}'$ is in a solution plane. WLOG, we can assume that the solution plane is

$H\big(+\mathbf{1},[[1,n']]\big)$, where $n' < n$ and $\text{sgn}(\mathbf{v}') = +\mathbf{1}$ (if not, we can simply gauge the polarity matrix and relabel the indices such that it is true). We first begin by showing that $\mathbf{C}(\mathbf{v}(\mathbf{x}',t)) = \mathbf{0}$ for $\forall t > 0$. To do so, it is sufficient to show that for all such $\mathbf{v}'$ (and arbitrary memory), the voltage flow field is positive, meaning that the trajectory will be pressed against the solution plane.

WLOG, we first focus only on the dynamics of v_1 influenced by clause j (assuming that $q_{1j} \neq 0$). The gradient influence is

$$G_{1j} = \frac{1}{2} q_{1j}(1 - q_{ij}v_i).$$

Note that the gradient-like term is non-positive only if $q_{1j} = -1$, which implies that $q_{ij}v_j = +1$ otherwise $C_j \neq 0$. In this case, we have $G_{1j} = 0$, therefore it is required that $G_{1j} \geq 0$ for all cases. For the rigidity term, we have

$$R_{1j} = \delta_{1\sigma_j} q_{1j} C_j(\mathbf{v}),$$

which is necessarily zero as $C_j(\mathbf{v}) = 0$. Therefore, all possible contributions to v_1 are non-negative, and this applies for $\forall i \in I$. This means that $\mathbf{C}(\mathbf{v}(t,\mathbf{x}')) = \mathbf{0}$ for $\forall t > 0$, then $\dot{\mathbf{x}}_\mathbf{s}(t) < \mathbf{0}$ and $\dot{\mathbf{x}}_\mathbf{l}(t) < \mathbf{0}$, so both memory variables will decay and terminate at 0 and 1 respectively.

The proof of the second part of this proposition is shown as Corollary A.7.2.1, immediately after we establish certain properties of the basin of attraction for our dynamics. □

Remark. *The proposition essentially states that once the voltage vector reaches a solution plane, then the dynamics will flow to a fixed point. On the other hand, if the voltage vector has not reached a solution plane yet, then the dynamics will continue to evolve (until it finds the solution). If the original 3-SAT problem is unsatisfiable, then the dynamics will continue to evolve forever.*

A.7 Basin of Attraction

From proposition A.3.1, we see that the 3-SAT problem is essentially solved once we have discovered a voltage assignment such that $\mathbf{C}(\mathbf{v}) < \frac{1}{2}$, and the dynamics can be terminated. However, in some cases, the implementation of this termination condition is perhaps not feasible, so we have to allow the dynamics to fully converge to a solution vector $\mathbf{v}_0$. In this case, it is necessary for us to determine the *basin of attraction* in which the dynamics are guaranteed to evolve towards the solution. We first formally define the basin of attraction as follows.

Definition A.7.1. *Given some flow field $F : \mathbb{R}^n \mapsto \mathbb{R}^n$, let $\mathbf{x}'$ be a fixed point of this field. We define the basin of attraction of $\mathbf{x}'$ as*

$$B(\mathbf{x}') = \{\mathbf{x}_0 \in \mathbb{R}^n \mid \lim_{t\to+\infty} \mathbf{x}(t, \mathbf{x}_0) = \mathbf{x}'\}.$$

Remark. *From the first part of proposition A.6.5, we see that every solution plane must contain a fixed point. We can then modify the above definition to solution plane as follows*

$$B(\mathbf{v}') = \{\mathbf{x}_0 \in \mathbb{R}^{n+2m} \mid \lim_{t\to+\infty} \mathbf{v}(t, \mathbf{x}_0) \in \bigcup_I H(\mathbf{v}', I)\},$$

where $H(\mathbf{v}', I)$ denotes the solution plane of $\mathbf{v}'$ corresponding to the proper index set I (see definition A.6.2). It is important to note that the basin of attraction of a solution vector is a subset of the full space, or $B(\mathbf{v}') \subseteq \mathbb{R}^{n+2m}$, even though the fixed points are defined in the voltage space $\mathbb{R}^n$. This is because the dynamics of the memory variables still affect the flow field of the voltages.

An object that will be often evoked in the following discussion is the orthant of the voltage space that contains a solution plane. To make the discussion easier, we shall refer to such orthant as a *solution orthant*.

Definition A.7.2 (Solution Orthant)**.** *Given a solution vector $\mathbf{v}' \in \mathbb{R}^n$ and a proper index set I*

(see definition A.6.2), we define its solution orthant to be

$$J(\mathbf{v}', I) = \{\mathbf{v} \in [-1,+1]^n \;\big|\; \forall i \in I,\, v_i v_i' \geq 0\}.$$

Furthermore, we define the restricted solution orthant to be

$$J'(\mathbf{v}', I) = \{\mathbf{v} \in [-1,+1]^n \;\big|\; \forall i \in I,\, v_i v_i' \geq 1 - 2\gamma\},$$

where $\gamma < \frac{1}{2}$ is the offset parameter defined in Eqs. (A.17).

Remark. *From the discussion in Section A.3.2, we see that a satisfiable 3-SAT problem can always be gauged such that the solution vector is $+\mathbf{1}$. This means that in our analysis, we can always assume that any solution orthant contains the positive orthant of $[-1,+1]^n$, as long as we guarantee that the dynamics are fully contained within the orthant.*

For better visualization, one can imagine the entire bounded space of the dynamics, O, as a hypercube. Then a solution orthant is simply a hyper-rectangle with some of its "edges" halved in such a way that it still contains a solution plane. A restricted solution orthant is constructed similarly except these edges are quartered instead. This can be described by the following containment relation

$$H(\mathbf{v}', I) \subset J'(\mathbf{v}', I) \subset J(\mathbf{v}', I) \subset O.$$

Lemma A.7.1. *Given a solution vector $\mathbf{v}'$ for which a proper index set I exists, we have*

$$\forall \mathbf{v} \in J'(\mathbf{v}', I), \qquad \mathbf{C}(\mathbf{v}) \leq \gamma.$$

On the other hand, given a vector $\mathbf{v} \in [-1,+1]^n$ such that $\mathbf{C}(\mathbf{v}) \leq \gamma$, let $\mathbf{v}' = \mathrm{sgn}(\mathbf{v})$ be the

corresponding solution vector. If there is a proper index set for this solution, then

$$\exists I, \qquad \mathbf{v} \in H(\mathbf{v}', I).$$

Proof. The proof follows trivially from definitions A.6.2 and A.7.2. □

Equipped with this lemma, we can now show that when a trajectory enters a restricted solution orthant with the fast variable being $\mathbf{x_f} = \mathbf{0}$, then it is guaranteed to converge to a solution plane, which further implies that it will evolve into a fixed point (see proposition A.6.5).

Theorem A.7.2 (Basin of Attraction)**.** *Let* $\mathbf{v}'$ *be a solution vector, then*

$$\left[\bigcup_I J'(\mathbf{v}', I)\right] \times \{0\}^m \times [1, +\infty)^m \subseteq B(\mathbf{v}').$$

Proof. It is sufficient to show

$$\forall I, \qquad J'(\mathbf{v}', I) \times \{0\}^m \times [1, +\infty)^m \subseteq B(\mathbf{v}').$$

WLOG, we let $\mathbf{v}' = +\mathbf{1}$ and the proper isolated index set be $I = [[1, n']]$, then $J'(\mathbf{v}') = J'(+\mathbf{1}) = [1-2\gamma, +1]^{n'} \times [-1, +1]^{n-n'}$, which we simply refer to as J' from here on. We first note that if $\mathbf{x_s} = \mathbf{0}$, then for $\forall \mathbf{v} \in J'$, we have $\dot{\mathbf{v}}(0) \geq 0$ (see the first line of Eqs. (A.17)). Furthermore, $\forall \mathbf{v} \in J'$, we have $\dot{\mathbf{x}}_\mathbf{f} \leq 0$ (which follows from the second line of Eqs. (A.17) and Lemma A.7.1). We first show, by contradiction, that for any point initialized in the supposed subset of the basin, then the evolution of each isolated component of the voltage vector must be weakly monotonous, or $\dot{v}_i(t) \geq 0$ for $\forall i \in [[1, n']]$ and $\forall t > 0$.

We let some initial point be $\mathbf{x}_0 = \{\mathbf{v}_0, \mathbf{x}_{f,0}, \mathbf{x}_{s,0}\} \in J' \times \{0\}^m \times [1, x_{\max}]^m$, and the solution trajectory be $\mathbf{x}(t)$. WLOG, we assume that $v_1(t)$ is not monotonously increasing, and is the first

voltage in time to violate the inequality $\dot{v}_1(t) \geq 0$. We denote this time to be

$$T = \inf\{t > 0 \mid \exists \dot{v}_1(t) < 0\}.$$

It is clear that $v_1(t) \in J'$ for $\forall t \in [0, T]$. In addition, it is required that $\mathbf{x_s}(T) \neq \mathbf{0}$. This is, however, only possible if $\exists t' \in [0, T]$ such that

$$\exists j \in [[1, m]], \quad \dot{x}_{s,j}(t') > 0.$$

But as $\mathbf{v}(t') \in J'$, the above condition is not possible. Therefore, by contradiction, we must have $\dot{\mathbf{v}}(t) \geq 0$ for $\forall t \geq 0$.

To complete the proof, it is sufficient to show that $\forall i \in I$, we have $\lim_{t\to\infty} v_i(t) = +1$. Again, we prove this by contradiction. We first assume that the statement is not true, then $\exists i \in I$ (WLOG let $i = n' - 1$), $\exists \varepsilon > 0$ such that $\lim_{t\to\infty} v_i(t) = 1 - \varepsilon$, and $\lim_{t\to\infty} \dot{v}_i(t) = 0$ (as $\mathbf{v}$ is monotonous). This means that there is a time T, after which v_i can no longer appear as the most satisfied literal in any clause. If this is not the case, then $\forall T$, $\exists t' > T$ such that $\dot{v}_i(t') = v_i(t')$, which is clearly not possible as the limits of the LHS and RHS converge to different values. As v_i is no longer the most satisfied literal in any clause, we can set its value arbitrarily in $[-1, +1]$, and the condition $\mathbf{C}(\mathbf{v}) \leq \boldsymbol{\gamma}$ will still remain true, as the clause energy of each clause only depends on the most satisfied literal (see Eq. (A.7)). From Lemma A.7.1, this implies that the restricted solution orthant should be $[+\frac{1}{2}, +1]^{n'-1} \times [-1, +1]^{n-n'+1}$ instead. However, the restricted solution orthant of a solution vector is unique given a proper index set I (see the remark of definition A.6.2), so we have a contradiction. Therefore, the dynamics must converge to a solution plane, and thus also to a fixed point by proposition A.6.5. □

Remark. *Note that this basin of attraction is a superset of the basin of attraction proven in*

another work [ERT11] using continuous dynamics for solving k-SAT problems. This means that the basin of attraction for our dynamics is larger, which is a desirable property for using our dynamics as a SAT solver.

Corollary A.7.2.1. *If* $\mathbf{x}'$ *is a fixed point of the memory dynamics given in Eqs. (A.17), then* $\mathbf{v}'$ *is in a solution plane.*

Proof. If $\mathbf{x}'$ is a fixed point, then clearly we require $C_j(\mathbf{v}') \leq \delta$ for $\forall j$, otherwise $x'_{l,j}$ will increase. And since $\delta < \gamma$, we have $C_j(\mathbf{v}') < \gamma$, meaning that $x'_{s,j} = 0$, otherwise $x'_{s,j}$ will decrease. Since $C_j(\mathbf{v}) \leq \delta < \frac{1}{2}$, $\text{sgn}(\mathbf{v}')$ is a solution vector (see Proposition A.3.1), and $\mathbf{x}' \in B(\text{sgn}(\mathbf{v}'))$ (see Theorem A.7.2). If $\mathbf{v}'$ is in a solution plane, then we are done; if not, then $\mathbf{x}'$ necessarily evolves to a solution plane by Theorem A.7.2, meaning that it cannot be a fixed point, creating a contradiction. Therefore, $\mathbf{v}'$ must already be in a solution plane to begin with. □

A.8 Dynamic Voltage Flow

Often times we are only interested in the convergent properties of the voltage dynamics, as they correspond directly to the state of Boolean variables. On the other hand, the evolution of the memory variables is important in influencing the trajectory of the voltages *indirectly* by changing the strength of the gradient-like and rigidity terms (see Eqs. (A.17)). It then makes sense to restrict our attention to only the components of the flow field that govern the dynamics of $\mathbf{v}$ directly, which we can denote as $F_{\mathbf{v}}$, and refer to as *reduced flow field* in the voltage space, or simply the *voltage flow*. It should be noted that this flow is not autonomous and is, in fact, dynamically governed by the memory. In this Section, we establish the tools needed to study such reduced flow field, which we will use to show certain properties such as the absence of periodic orbits (see Section A.9) in the voltage space. For the remainder of this material, we shall assume that the full flow field is always Lipschitz continuous to simplify discussion, since we have seen in Section A.6.1 that the existence of measure-zero discontinuities does not alter significantly

the behavior of our dynamics. Often times, we will focus on the simplified dynamics as given in Eq. (A.21) and assume continuity for **G** and **C**.

A.8.1 Reduced Flow

In this Section, we will first factor the full phase space into the *reduced* space and the *auxiliary* space, which will allow us to formalize the notion of a reduced flow field. Visually, the reduced flow field can be viewed as the full flow field "projected" onto a subspace. We proceed with the following series of definitions.

Definition A.8.1. *Let X be a set and $Y = \prod_{j=1}^{m} Y_j$ be a product space. Given any mapping $F : X \mapsto Y$, we define the decomposition of F as $F_j : X \mapsto Y_j$ for $\forall j \in [[1,m]]$ such that*

$$\forall x \in X, \qquad F(x) = \big(F_1(x), F_2(x), ..., F_m(x)\big).$$

Definition A.8.2. *Let $F = (F_1, F_2) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n \times \mathbb{R}^m$ be a flow field, and $\mathbf{x}(t, \mathbf{x}_0)$ be a trajectory under this flow field. If we denote $\mathbb{R}^n$ as the reduced space and $\mathbb{R}^m$ as the auxiliary space, then we define the reduced trajectory and the auxiliary trajectory, $\mathbf{x}_1(t, \mathbf{x}_0) \in \mathbb{R}^n$ and $\mathbf{x}_2(t, \mathbf{x}_0) \in \mathbb{R}^m$, such that*

$$\mathbf{x}(t, \mathbf{x}_0) = \big(\mathbf{x}_1(t, \mathbf{x}_0), \mathbf{x}_2(t, \mathbf{x}_0)\big).$$

Definition A.8.3 (Reduced Flow)**.** *Let $F = (F_1, F_2) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n \times \mathbb{R}^m$ be a flow field, and $\mathbf{x}(t, \mathbf{x}_0)$ be a trajectory under this flow field. For a given initial point $\mathbf{x_0}$, we construct the reduced flow field $F_r : \mathbb{R} \times \mathbb{R}^n$ such that*

$$\forall t \geq 0, \qquad F_r(t, \mathbf{x}_{0,1}) = F_1\big(\mathbf{x}_{0,1}, \mathbf{x}_2(t, \mathbf{x}_0)\big).$$

Remark. *It can be easily verified that the reduced flow field is well defined at any given time. Visually, if we view the reduced space as a hyperplane that "cuts" the full flow field, then the*

reduced flow field is simply the "cross section" of the field in the plane. As the auxiliary variables evolve in time, the plane will move in the auxiliary space, or simply some direction orthogonal to the plane, thereby changing the cross section. We then see that the reduced flow field is effectively a dynamic flow field governed by the auxiliary trajectory.

A.8.2 Flow Kernel and Complement

In this subsection, we will relate the dynamics of the reduced flow field to the auxiliary trajectory explicitly. We will focus specifically on the case where the reduced flow field is linear in the auxiliary variables, with the simplified memory dynamics in Eq. (A.21) as an example. In particular, at every point in the reduced space, the auxiliary space can be factored into two subspaces, one of them in which the auxiliary trajectory can evolve without affecting the reduced flow field. We refer to this subspace as the *flow kernel* of the reduced flow field at that point, and the other factor subspace as the *flow complement*. We formally define the two subspaces as follows:

Definition A.8.4 (Flow Kernel)**.** *Let $F = (F_1, F_2) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n \times \mathbb{R}^m$ be a vector field. If $F_1(\mathbf{x}_1, \mathbf{x}_2)$ is linear in $\mathbf{x}_2$, then we define*

$$K_{F_1}(\mathbf{x}_1) = \{\mathbf{x}_2 \in \mathbb{R}^m \;\big|\; F_1(\mathbf{x}_1, \mathbf{x}_2) = 0\},$$

as the flow kernel of F_1 at $\mathbf{x}_1$.

Remark. *Clearly, the flow kernel is a vector space. In fact, given any fixed $\mathbf{x}_1$, the operation $F_1(\mathbf{x}_1, \mathbf{x}_2)$ can be regarded as a mapping from $\mathbb{R}^m$ to $\mathbb{R}^n$ via an $n \times m$ matrix, with its kernel being the flow kernel. Let the rank of the matrix be $n' \leq n$. If $n' \geq m$, then clearly the kernel is trivial. On the other hand, if $n' < m$, then the dimension of the kernel is $m - n'$ by the rank-nullity theorem. Hard 3-SAT instances generally are at clause ratios near 4, meaning that $m \approx 4n$ (if we*

let the voltage space be the reduced space), and the kernel is generally non-trivial.

Definition A.8.5 (Flow Complement)**.** *Let $F = (F_1, F_2) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n \times \mathbb{R}^m$ be a vector field. We refer to the orthogonal complement of the flow kernel at point $\mathbf{x_1}$,*

$$J_{F_1}(\mathbf{x}_1) = \{\mathbf{x}_2 \in \mathbb{R}^m \mid \forall \mathbf{x}_2' \in K_{F_1}(\mathbf{x}_1), \quad \mathbf{x}_2 \cdot \mathbf{x}_2' = 0\},$$

as the flow complement of F_1 at $\mathbf{x}_1$.

Remark. *For any fixed $\mathbf{x}_1$, if the domain of $F_1(\mathbf{x}_1, \mathbf{x}_2)$ is restricted to $J_{F_1}(\mathbf{x}_1)$, then the mapping is invertible. In other words, there is a bijection from every configuration of the auxiliary variables in the flow complement to every possible flow vector. In some sense, the component of the auxiliary variable in the flow complement space is the only relevant component generating the reduced flow.*

Definition A.8.6 (Auxiliary Relevance)**.** *Let $F = (F_1, F_2) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n \times \mathbb{R}^m$ be a vector field. Given $\mathbf{x}_1 \in \mathbb{R}^n$ and $\mathbf{x}_2 \in \mathbb{R}^m$, we refer to the projection of $\mathbf{x}_2$ to $K_{F_1}(\mathbf{x}_1)$ as the irrelevant component, and the projection to $G_{F_1}(\mathbf{x}_1)$ as the relevant component, which we denote as $\mathbf{x}_2^*$.*

Remark. *Given a reduced flow field that is linear in the auxiliary variables, it can be shown that the time derivative of the field is zero at time t and location $\mathbf{x}_{0,1}$, if and only if the auxiliary variable evolves in the flow kernel of F_1, or*

$$\dot{\mathbf{x}}_2(t, \mathbf{x}_0) \in K_{F_1}(\mathbf{x}_1).$$

Equivalently, this means that the time derivative of the relevant component of $\mathbf{x}_2$ must be zero, or

$$\dot{\mathbf{x}}_2^*(t, \mathbf{x}_0) = 0.$$

A.8.3 Unstable Non-solution Fixed Points

In proposition A.6.5, it was shown that every fixed point in the full phase space $\mathbb{R}^{n+2m}$ must correspond to a solution of the 3-SAT problem. However, this still leaves open the possibility of the existence of fixed points in the voltage space that correspond to a non-solution. Most of the time, when the dynamics fall into such fixed points, the memory breaks this fixed point by reweighing the clause functions, thereby evolving the reduced flow vector to a non-zero value, effectively freeing the voltage dynamics. However, in very rare instances, the memory variables may evolve in the flow kernel, in which case the voltages may be permanently trapped. Here, we show the unlikeliness of being trapped in such fixed points in *general*, and the *instability* of the gradient-like influence near fixed points.

For simplicity, we focus on the simplified memory dynamics as given in Eq. (A.21): [12]

$$\dot{\mathbf{v}} = \mathbf{G}(\mathbf{v})\mathbf{x}$$
$$\dot{\mathbf{x}} = \alpha\mathbf{C}(\mathbf{v}).$$

We here temporarily relax the specific forms of functions $\mathbf{C}$ and $\mathbf{G}$ (Eqs. (A.7) and (A.18) respectively), and simply require they be general and non-singular. A voltage fixed point means that $\dot{\mathbf{v}} = \mathbf{0}$, implying that the memory must be in the flow kernel, or $\mathbf{x} \in K(\mathbf{v})$. If the condition $\dot{\mathbf{v}} = \mathbf{0}$ is to be held in time (or $\ddot{\mathbf{v}} = \mathbf{0}$), then the memory must also evolve in the flow kernel, or $\dot{\mathbf{x}} \in K(\mathbf{v})$. Equivalently, $\mathbf{G}(\mathbf{v}) \cdot \mathbf{C}(\mathbf{v}) = \mathbf{0}$. The LHS is simply a $\mathbb{R}^n \mapsto \mathbb{R}^n$ mapping, so the preimage of $\mathbf{0}$ consist of finitely many points in *general*, and they constitute a measure-zero set in $\mathbb{R}^n$. This shows the unlikeliness of the dynamics being trapped in a non-solution fixed point.

[12] If we were to extend the analysis of this subsection to the full memory dynamics (by including the rigidity and fast memory dynamics), then the RHS to $\dot{\mathbf{v}}$ can be decomposed into two terms, one quadratic in $\mathbf{x}$ and the other being only dependent on $\mathbf{v}$. The equation $\dot{\mathbf{v}} = \mathbf{0}$ would still be a polynomial equation for $\mathbf{x}$, and the solution space of $\mathbf{x}$ can be similarly decomposed into a hyperface defined by the corresponding algebraic variety and its complement, and the analysis in this subsection can be easily extended for the full memory dynamics as well by considering the local tangent space.

To show that the gradient-like influence is unstable, we first note that a Jacobian element of the gradient-like term can be written as

$$\mathcal{J}_{ij} = \sum_k x_k \partial_{v_i} G_{jk}(\mathbf{v}) = \frac{1}{2} \sum_k x_k q_{jk} \partial_{v_i} \min_{\{j' \neq j \,|\, q_{j'k} \neq 0\}} (1 - q_{j'k} v_{j'}),$$

where the last equality is from Eq. (A.18), and derivations across the discontinuous hyperplanes are neglected. In this form, it is clear that the diagonal elements of the Jacobian are zero, or $\mathcal{J}_{ii} = 0$ for $\forall i$. To see this, we simply note that $\partial_{v_i} v_{j'} = \delta_{ij'}$, and the condition $i \neq j'$ imposed by the min function. This means that the trace of the Jacobian is zero, meaning that any fixed point cannot be stable (otherwise the Jacobian would necessarily be negative in the real component of the trace).

A.9 Non-periodicity of Dynamics

In dimensions greater than 2, a dissipative system[13] may admit periodic orbits. Therefore, we shall show the absence of periodic orbits explicitly in this Section. This result directly precludes the possibility of chaos (see Section A.9.4). We formulate the proof of non-periodicity on the voltage space by making use of the formalism developed in Section A.8. Note that showing the absence of periodic orbits in the full state space (voltages plus memories) is not sufficient for our purpose as it does not preclude the existence of periodic orbits in the reduced voltage space, which is directly relevant to the solution of the 3-SAT problem. For analytic convenience, we shall assume all mentioned fields in this Section is sufficiently well-behaved (i.e., Lipschitz continuous in space and continuous in time) such that it admits a unique classical solution for all initial values.

[13] See Section A.10 for a detailed discussion of the dissipativeness of the memory dynamics.

A.9.1 Generalized Periodicity

As the voltage dynamics by itself is not autonomous (since it is memory dependent), we first have to construct a non-standard definition of periodicity for dynamic fields that suffices in the context of optimization. In general, a dynamic field admits periodic orbits of non-constant periods. We first recall that the classical definition of periodicity for static fields is given as follows, and generalize this definition for dynamic fields.

Definition A.9.1 (Regular Periodic Orbit)**.** *Let* $\mathbf{x} : [0, +\infty) \mapsto \mathbb{R}^n$ *be a trajectory with initial value* $\mathbf{x}_0$*. The trajectory is said to be periodic if* $\exists T > 0$ *such that*

$$\forall t \geq 0, \qquad \mathbf{x}(t+T, \mathbf{x_0}) = \mathbf{x}(t, \mathbf{x_0}).$$

The periodic orbit of $\mathbf{x}_0$ *is*

$$\gamma_x = \{\mathbf{x}(t, \mathbf{x_0}) \;\big|\; t \in [0, T)\},$$

and the period of this orbit is T*.*

Remark. *It is fairly easy to show the following*

$$\begin{aligned} \gamma_x &= \{\mathbf{x}(t) \;\big|\; t \in [0, T)\} \\ &= \{\mathbf{x}(t) \;\big|\; t \geq 0\} = \gamma_x^+, \end{aligned}$$

meaning that the periodic orbit is also the maximal positive orbit of $\mathbf{x}_0$*, which makes sense because the trajectory cannot escape the periodic orbit even given infinite time. This property generalizes naturally for dynamic fields.*

Definition A.9.2 (Speed)**.** *Let* $\mathbf{x} : [0, +\infty) \mapsto \mathbb{R}^n$ *be some trajectory with initial value* $\mathbf{x}_0$*. If the trajectory is everywhere differentiable in time, we define the velocity along the trajectory to be*

$$\dot{\mathbf{x}}(t, \mathbf{x}_0),$$

and the speed to be

$$s(t, \mathbf{x}_0) = ||\dot{\mathbf{x}}(t, \mathbf{x}_0)||.$$

Remark. *Clearly, the velocity and speed of the trajectory is also periodic with the same period as the trajectory itself. If the trajectory is governed by the flow field F, then the period of the orbit is given by the following contour integral*

$$T = \oint_{\gamma} \frac{||d\mathbf{x}||}{||F(\mathbf{x})||}.$$

This integral is well-defined for a static flow field, but it is no longer well defined if F is explicitly time dependent, in which case the period may be time-dependent as well.

Definition A.9.3 (General Periodic Orbit)**.** *Given a time-dependent flow field $F : \mathbb{R} \times \mathbb{R}^n \mapsto \mathbb{R}^n$, a general periodic orbit is said to exist for $\mathbf{x_0}$ if $\exists T$ such that*

$$\mathbf{x}(T, \mathbf{x_0}) = \mathbf{x_0}.$$

Let the periodic orbit be $\gamma_x = \{\mathbf{x}(t) \,|\, t \in [0, T)\}$, then it is required that $\mathbf{x}(t, 0) \in \gamma_x$ for $\forall t > 0$. Furthermore, $\exists (t_1, t_2) \in \{(s_1, s_2) \in [0, +\infty)^2 \,|\, s_1 \neq s_2\}$ such that

$$\mathbf{x}(t_1) = \mathbf{x}(t_2) \quad \wedge \quad \dot{\mathbf{x}}(t_1), \dot{\mathbf{x}}(t_2) \neq 0$$

For a given time t, we let

$$T'(t) = \inf\{t' > t \;\big|\; \mathbf{x}_1(t', \mathbf{x}_0) = \mathbf{x}_1(t, \mathbf{x}_0)\},$$

then the period at time t is given as $T(t) = T'(t) - t$. If $T(t)$ is a constant in time, then the periodic orbit is said to be regular; otherwise, the periodic orbit is said to be irregular.

Remark. *Essentially, a general periodic orbit is a closed trajectory which contains the maximal positive solution. Furthermore, there must be a point and its neighborhood on the orbit which the trajectory visits two separate times, with the period simply being the time duration until the next visit. Technically, the period can be zero if the flow field is zero at that particular time and point and infinite if the trajectory never revisits the point, but there must be at least one point in time where the period is positive finite.*

Lemma A.9.1. *If a dynamic flow field* $F : \mathbb{R} \times \mathbb{R}^n \mapsto \mathbb{R}^n$ *admits an irregular periodic orbit, then* $\exists \mathbf{x}_0$ *such that* $\exists t_1 \geq 0$, $\exists t_2 \in \{t' > t_1 \mid \mathbf{x}(t', \mathbf{x_0}) = \mathbf{x}(t_1, \mathbf{x_0})\}$,

$$\exists k \neq 1 \qquad F(t, \mathbf{x}(t_2)) = kF(t, \mathbf{x}(t_1)) \neq 0.$$

Proof. The proof is omitted. See remark instead. □

Remark. *Essentially, there must be at least one point on the periodic orbit where the dynamic flow field align (or anti-align) with itself at two separate times. This is clearly required so that the trajectory can "revisit" the orbit in the neighborhood of that point.*

Corollary A.9.1.1 (Change in Relevant Component)**.** *Given a static flow field* $F = (F_1, F_2) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n \times \mathbb{R}^m$*, if* $F_1(\mathbf{x}_1, \mathbf{x}_2)$ *is linear in* $\mathbf{x}_2$ *and an irregular periodic orbit* $\boldsymbol{\gamma}$ *exists in* $\mathbb{R}^n$*, then* $\exists \mathbf{x}_0 \in \mathbb{R}^{n+m}$ *such that* $\exists t_1 \geq 0$, $\exists t_2 \in \{t' > t_1 \mid \mathbf{x_1}(t', \mathbf{x_0}) = \mathbf{x_1}(t_1, \mathbf{x_0})\}$,

$$\exists k \neq 1 \qquad \mathbf{x_2}^*(t, \mathbf{x_0}) = k\mathbf{x_2}^*(t, \mathbf{x_0}) \neq 0.$$

where $\mathbf{x}_2^*(t, \mathbf{x}_0)$ *is the relevant component of* $\mathbf{x}_2$ *at time t as defined in Definition A.8.6.*

Proof. The proof follows directly from Definition A.8.6 and Lemma A.9.1.1. □

Remark. *In terms of the relevant component of the auxiliary variable, the periodic orbit is*

regular if and only if

$$\forall t \geq 0, \exists T > 0 \qquad \mathbf{x}_2^*\big(t+T, \mathbf{x}_0\big) = \mathbf{x}_2^*(t, \mathbf{x}_0).$$

A.9.2 Absence of Irregular Periodic Orbits

In the previous subsection, we have seen that a periodic orbit under a dynamic field can be categorized as either being regular or irregular. To show the absence of periodic orbits, we treat the two cases separately, as they require different proof techniques. In this subsection, we focus on the irregular case, which requires the physical notion of speed as defined in definition A.9.2; we treat the regular case in the next subsection, by formulating the problem in the geometric context of hypersurface intersections.

For the sake of simplicity, we again focus on the simplified dynamics as given in Eq. (A.21)[14],

$$\dot{\mathbf{v}} = \mathbf{G}(\mathbf{v})\mathbf{x}$$

$$\dot{\mathbf{x}} = \alpha\mathbf{C}(\mathbf{v}).$$

We require the functions $\mathbf{C}$ and $\mathbf{G}$ to be everywhere differentiable[15] (which automatically guarantees Lipscthiz continuity). This also guarantees that any image of a compact set in $\mathbb{R}^n$ is bounded above in norm. For $\mathbf{C}$, we can assume that it is bounded below in norm also, otherwise $\mathbf{C} = 0$ implies that the trajectory is in a solution plane in which case it must converge to a fixed point

[14]The rigidity influence is negligible in the periodicity analysis as the dynamics is dominated by the gradient-like term when the system is continuously in an unsatisfied state, which is clearly the case when the dynamics is trapped in a periodic orbit.

[15]Note that the actual gradient-like term $\mathbf{G}$ defined in Eq. (A.18) is everywhere differentiable except at certain hyperplanes which constitute a measure-zero set in the voltage space (see section A.6.1). It is easy to see that the presence of these hyperplanes will not affect the periodicity analysis.

(see Proposition A.6.5)[16]. To make the definition of speed (see Definition A.9.2) useful, we first have to formally define the generalized concept of *location* for trajectories governed by a dynamic field.

Definition A.9.4 (Location)**.** *Let* $\mathbf{x}(t,\mathbf{x_0})$ *be a classical solution to a flow field* $F:\mathbb{R}\times\mathbb{R}^n\mapsto\mathbb{R}^n$*, then* $\forall t'\in\{t\in\mathbb{R}\,|\,\mathbf{x}(t,\mathbf{x_0})\neq 0\}$*, there is* $\exists\delta t>0$ *such that a unique isometry* $\varphi:\mathbf{x}\big([t'-\delta t,t'+\delta t],\mathbf{x_0}\big)\mapsto\mathbb{R}$ *exists locally. For* $\forall t\in[t'-\delta t,t+\delta t]$*, we refer to* $u(t)=\varphi(t)$ *as the location on the trajectory around time* t'*.*

Remark. *The location is quite literally the location on the real line if we unwind the trajectory locally to a straight line. As long as the trajectory keeps moving "forward" in some time interval, then the mapping* φ *is bijective, meaning that there is a one-to-one correspondence between time and location. This is why the condition* $\mathbf{x}(t,\mathbf{x_0})\neq 0$ *is required locally.*

Lemma A.9.2. *In a time interval in which location can be defined, the speed is differentiable with respect to location in the interval. In other words, the mapping* $s\circ u^{-1}$ *is locally differentiable.*

Proof. First of all, we have $s(t)=||\dot{\mathbf{v}}(t)||\neq 0$ in the time interval, and we note that

$$\ddot{\mathbf{v}}=\frac{d}{dt}\Big(\mathbf{G}(\mathbf{v})\mathbf{x}\Big)=\dot{\mathbf{v}}\mathbf{G}'(\mathbf{v})\mathbf{x}+\mathbf{G}(\mathbf{v})\dot{\mathbf{x}},$$

which is well-defined as $\mathbf{G}$ is everywhere differentiable, meaning that $s(t)$ is differentiable with respect to t (as long as $s(t)\neq 0$). Furthermore, we note that $u'(t)=s(t)$, meaning that u is also differentiable with respect to time. Since $s(t)\neq 0$, the inverse u^{-1} is differentiable in the interval as well. Therefore, $s\circ u^{-1}$ is differentiable in the interval, as the composition of two differentiable mappings. □

Theorem A.9.3. *An irregular orbit does not exist in the voltage space.*

[16] In fact, it can be assume that $|\mathbf{C}(\mathbf{v})|\geq\delta$ for the full equations by Proposition A.7.2.

Proof. We prove this by contradiction, by assuming that an irregular orbit does exist. Then by Lemma A.9.1, there is $\exists \mathbf{v_0}$ such that $k\dot{\mathbf{v}}(t_1) = \dot{\mathbf{v}}(t_2) \neq 0$, where $t_2 \neq t_1$, $\mathbf{v}(t_1) = \mathbf{v}(t_2) = \mathbf{v_0}$, and WLOG $k \in (0,1)$. We let u_0 be the location of $\mathbf{v_0}$, δu be the infinitesimal change in location from u_0. Furthermore, we let s_1 be the speed at time t_1, and δs be the change in speed with respect to δu (which is well-defined as shown in Lemma A.9.2). If we let $\delta\theta$ be the change in direction, then we have the following equality

$$\begin{aligned}(s_1+\delta s)^2 + s_1^2 - 2s_1(s_1+\delta s)\cos(\delta\theta) &= ||\dot{\mathbf{v}}'(u_0)||^2\delta u^2 \\ \iff \quad s_1^2\delta\theta^2 + \delta s^2 &= \left\|\mathbf{G}'(u_0)\mathbf{x}(t_1) + \frac{\alpha}{s_1}\mathbf{G}(u_0)\mathbf{C}(u_0)\right\|^2 \delta u^2,\end{aligned}$$

where we discarded third order terms on the LHS and applied Eq. (A.21) and $u'(t) = s(t)$ on the RHS. Rearranging the terms gives us

$$\big(s'(u_0)\big)^2 = \left\|\mathbf{G}'(u_0)\mathbf{x}(t_1) + \frac{\alpha}{s_1}\mathbf{G}(u_0)\mathbf{C}(u_0)\right\|^2 - s_1^2\big(\theta'(u_0)\big)^2.$$

If we let the speed at time t_2 be s_2 (with $s_2 = ks_1$), then the above relation will hold similarly. We can assume that $s'(u_0) = 0$ at t_2, which is justified as s is bounded and differentiable everywhere. Since $s_1 > s_2$, and $\{\mathbf{x}, \mathbf{C}, \mathbf{G}\}$ are everywhere differentiable, we must have $\big(s'(u_0)\big)^2 < 0$ at time t_1 by the above relationship, which is clearly impossible. Therefore, an irregular orbit cannot exist.

□

A.9.3 Absence of Regular Periodic Orbits

In the previous subsection, we showed the absence of irregular orbits, so if a periodic orbit were to exist in the voltage space, it must be a regular periodic orbit. In this Section, we show that the existence of a periodic orbit is also absent in general. The problem can be

described geometrically where a regular orbit can be described as the intersection between two low dimensional hypersurfaces in a high-dimensional space, which cannot occur if the two surfaces are in general positions.

Theorem A.9.4. *In general, a periodic orbit does not exist in the voltage space.*

Proof. In theorem A.9.3, we showed the absence of irregular periodic orbits in the voltage space, it is then sufficient to show that a regular periodic orbit is absent as well.

If a regular orbit were to exist in the voltage space, then we can denote its period as T, and its initial point as $\mathbf{x}_0 \in \mathbb{R}^{n+m}$. The change in the memory over a period from time t is then given by

$$\begin{aligned}
&\mathbf{x}(t+T,\mathbf{x}_0) - \mathbf{x}(t,\mathbf{x}_0) \\
=&\int_t^{t+T} \dot{\mathbf{x}}(s,\mathbf{x}_0)\, ds \\
=&\alpha \int_t^{t+T} \mathbf{C}\big(\mathbf{v}(s,\mathbf{x}_0)\big)\, ds \\
=&\alpha \int_0^{T} \mathbf{C}\big(\mathbf{v}(s,\mathbf{x}_0)\big)\, ds, \qquad \forall t \geq 0,
\end{aligned}$$

where in the last equality, we used the periodicity of $\mathbf{v}$ to remove the explicit dependency on t in the integral bounds. This allows us to simply set the result as some constant vector $\mathbf{K}$ that is constant in time.

Clearly, $\mathbf{K}$ must be in the flow kernel of the reduced flow field $\forall t \geq 0$ (otherwise the velocity would not be the same after a period), which implies

$$\forall \mathbf{v} \in \gamma, \qquad \mathbf{G}(\mathbf{v})\,\mathbf{K} = \mathbf{0}. \tag{A.23}$$

It can be assumed that any sensible matrix $\mathbf{G}$ dictating the evolution of the voltages must coincide

with the polarity matrix $\mathbf{Q}$ exactly in its nonzero elements, so $\mathbf{G} : \mathbb{R}^n \mapsto \mathbb{R}^{3m}$ as there are m clauses and 3 literals per clause, which gives us an injective mapping. Furthermore, it can be assumed that the mapping is C^∞ diffeomorphic and general so that the image of $\mathbb{R}^n$ is a smooth n-dimensional hypersurface at general position in $\mathbb{R}^{3m}$. On the other hand, condition (A.23) is a system of m linear equations, so the set of all matrices (with nonzero elements matching the polarity matrix) solving the system for a given $\mathbf{K}$ forms a $3m - m = 2m$ dimensional hyperplane in $\mathbb{R}^{3m}$, which is also in general position as $\mathbf{K}$ is general. The n-dimensional hypersurface generated by the voltages and the $2m$ dimensional hyperplane do not intersect if they are in general position, as

$$2m + n < 3m,$$

where we have assumed $n < m$ (see the remark of definition A.8.4). □

Remark. *Note that the dimension of the surface containing the periodic orbit must be at least 2, which means that the intersection of the two surfaces in $\mathbb{R}^{3m}$ must be at least 2 dimensional as well, and this makes the existence of periodic orbits even less likely. Even assuming that $n > m$, meaning that the intersection of the two surfaces is non-trivial, the existence of a periodic orbit in the voltage space is still unlikely. We require an initial memory value that generates a reduced flow field that guarantees the containment of the voltage trajectory completely in the intersection, which does not exist in general.*

A.9.4 Absence of Chaos

Devaney's definition of chaos [Dev92] requires a dense set of periodic orbits and topological transitivity. We have shown in the previous Section that periodic orbits are not supported by the dynamics defined by Eqs. (A.17), and this directly precludes the existence of chaos. We can then state the following:

Corollary A.9.4.1 (Absence of Chaos)**.** *The voltage dynamics are non-chaotic.*

Remark. *Although not required to show the absence of chaos, we also note that the voltage dynamics are not topologically transitive if a solution of the 3-SAT problem exists which implies the existence of fixed points (See Proposition A.6.5). This is because the dynamics are confined in a compact positive invariant set O with nonempty interior (see Section A.6.2) and there is at least one fixed point in that set with its ω-limit set that is not O.*

A.10 Dissipativeness

A rather important property of the memory dynamics is dissipativeness. In other words, the measure (or volume) of an initial set contracts under the flow field, eventually evolving to a measure zero set. To show dissipativeness for well-behaved (everywhere differentiable) vector fields, it is sufficient to show that the divergence is negative everywhere. However, our dynamics are governed by a discontinuous flow field, so we have to carefully account for the regions of discontinuities (see Section A.6.1).

A.10.1 Preliminaries

Before we discuss the dissipative property of the memory dynamics, we first have to formally define the notion of dissipativeness for a continuous dynamical system.

Definition A.10.1 (Dissipativeness)**.** *Given a vector field $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ that admits a positive solution, we let the corresponding time mapping be $T_s : \mathbb{R}^n \mapsto \mathbb{R}^n$ for $\forall s \geq 0$. Let $\Omega_0 \subseteq \mathbb{R}^n$ be a domain of nonzero measure, then $\forall s \geq 0$, we denote $\Omega(s, \Omega_0)$ as the following*

$$\Omega(s, \Omega_0) = T_s(\Omega_0) = \{T_s(\mathbf{x}_0) \,|\, \mathbf{x_0} \in \Omega_0\},$$

and the measure $\mu(s,\Omega_0)$ as the following

$$\mu(s,\Omega_0) = \mu\big(\Omega(s,\Omega_0)\big).$$

If the following is true

$$\forall \Omega_0, \quad \forall s > 0, \quad \mu(s,\Omega_0) < \mu(0,\Omega_0),$$

then the system is said to be dissipative. If the last inequality is not strict, then the system is said to be weakly dissipative.

Remark. *If $\mu(s,\Omega_0)$ is everywhere differentiable in s, then it is possible for us to quantify the rate of volume contraction as the following forward time derivative*

$$\forall s \geq 0, \qquad \dot{\mu}(s,\Omega_0) = \lim_{t\to 0^+} \frac{1}{t}\big(\mu(s+t,\Omega_0) - \mu(s,\Omega_0)\big).$$

An equivalent definition of a dissipative system would then be the following

$$\forall \Omega_0, \quad \dot{\mu}(0,\Omega_0) < 0,$$

meaning that any initial domain must continually shrink in time. For the sake of clarity, we can discard the trivial argument $s = 0$ and simply write $\dot{\mu}(\Omega_0) = \dot{\mu}(0,\Omega_0)$ from here on.

It is well known that a bounded domain can be approximated[17] as the union of regular domains [Mun14] (see Definition A.5.2). Therefore, to show that a system is dissipative, it is sufficient to show that the volume of any regular domain contracts under the flow field. In the case where the flow field is in C^1, the mapping T_s is diffeomorphic for $\forall s > 0$, meaning that the shape of the boundary will be preserved (being always diffeomorphic to a sphere), allowing us to

[17] Here, we are speaking of approximation in the measure-theoretic sense. In other words, the measure of the domain and the measure of its approximation are the same.

make use of the following Lemma.

Lemma A.10.1. *Given a vector field $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ differentiable everywhere, the system is dissipative if the following is true*

$$\forall \mathbf{x} \in \mathbb{R}^n, \qquad \nabla \cdot F(\mathbf{x}) < 0.$$

Proof. The proof follows directly from divergence theorem

$$\dot{\mu}(\Omega) = \int_{\partial\Omega} \big(F(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x})\big)\, dA = \int_{\Omega} \big(\nabla \cdot F(\mathbf{x})\big)\, dV < 0,$$

where $\Omega \subseteq \mathbb{R}^n$ is any smooth domain. This implies that F is dissipative.

□

Remark. *The converse of Lemma A.10.1 is almost true, in the sense that if the vector field F contains regions of positive*[18] *divergence, then the system cannot be dissipative. To show this, we assume $\exists \mathbf{x} \in \mathbb{R}^n$ such that $\nabla \cdot F(\mathbf{x}) > 0$, then it is clear that the region where the divergence is positive*

$$D = \{\mathbf{x} \in \mathbb{R}^n \,|\, \nabla \cdot F(\mathbf{x}) > 0\}$$

is an open set. This means that for any $\mathbf{x}_0 \in D$, $\exists \varepsilon > 0$ such that the open ball $B_\varepsilon(\mathbf{x}_0) \subset D$, and integrating the divergence over the open ball gives

$$\int_{B_\varepsilon(\mathbf{x}_0)} \big(\nabla \cdot F(\mathbf{x})\big)\, dV > 0,$$

implying that $\dot{\mu}\big(B_\varepsilon(\mathbf{x}_0)\big) > 0$, meaning that the system cannot be dissipative.

[18] Note that it is not sufficient that the divergence be non-negative, as it is possible for the divergence to be zero, forming a closed set. This is why the converse of Lemma A.10.1 is not strictly true.

The analysis in this subsection assumes that the field is differentiable everywhere. In the next subsection, we generalize this analysis to fields that are differentiable everywhere except at certain hyperplanes. This class of fields contains the flow field of the memory dynamics (see Section A.6.1).

A.10.2 Dissipativeness of Memory Dynamics

Recall that the ODE governing the memory dynamics is given in Eqs. (A.17) as

$$\dot{v}_i = \sum_{j=1}^{m} \Big\{ \frac{1}{2} x_{l,j} x_{s,j} q_{ij} \min_{\{i' \neq i \,|\, q_{i'j} \neq 0\}} (1 - q_{i'j} v_{i'}) + (1 + \zeta x_{l,j})(1 - x_{s,j}) \delta_{i\sigma_j} q_{ij} C_j(\mathbf{v}) \Big\},$$

$$\dot{x}_{s,j} = \beta\big(x_{s,j} + \varepsilon\big)\big(C_j(\mathbf{v}) - \gamma\big),$$

$$\dot{x}_{l,j} = \alpha\big(C_j(\mathbf{v}) - \delta\big).$$

In Section A.6.1, we argued that the memory flow field is separated into continuous regions by hyperplanes. The divergence cannot be defined at the hyperplanes as the field is discontinuous, so we restrict the divergence analysis to a domain where the field is in C^1.

The divergence of the flow field in the voltage space[19] is

$$\begin{aligned}
\nabla_{\mathbf{v}} \cdot F_{\mathbf{v}} =& \nabla_{\mathbf{v}} \Big(\mathbf{G}(\mathbf{v}) (\mathbf{x_l} * \mathbf{x_s}) \Big) + \nabla_{\mathbf{v}} \Big(\mathbf{G}(\mathbf{v}) \big((1 + \zeta \mathbf{x_s}) * (1 - \mathbf{x_l}) \big) \Big) \\
=& \sum_{ij} x_{l,j} x_{s,j} \partial_{v_i} G_{ij}(\mathbf{v}) + \sum_{ij} (1 + \zeta x_{l,j})(1 - x_{s,j}) \partial_{v_i} R_{ij}(\mathbf{v}) \\
=& \sum_{ij} (1 + \zeta x_{l,j})(1 - x_{s,j}) q_{ij} \partial_{v_i} \big(\delta_{i\sigma_j} C_j(\mathbf{v}) \big),
\end{aligned}$$

where the divergence of the gradient-like term is zero because the element G_{ij} never depends on v_i (see Eq. (A.18)). The expression for the divergence of the rigidity term can be further

[19] The reason why we only care about the dissipativeness in the voltage space is because it is directly relevant to the convergence of the 3-SAT solution search. It is possible for the solver to be efficient even if the memory is non-dissipative.

simplified if we realize that

$$\sum_i q_{ij}\delta_{i\sigma_j}\partial_{v_i}C_j(\mathbf{v}) = -\frac{1}{2}\sum_i q_{ij}^2\delta_{i\sigma_j} = -\frac{1}{2}q_{\sigma_j j}^2 = -\frac{1}{2}.$$

And the divergence expression reduces to

$$\nabla_{\mathbf{v}}\cdot F_{\mathbf{v}} = -\frac{1}{2}\sum_j(1+\zeta x_{l,j})(1-x_{s,j}) < 0,$$

meaning that the voltage is *dissipative* at any point where the field is continuous.

It is easy to study how the addition of discontinuous hyperplanes affects the dissipativeness of the voltages. For the sake of simplicity, we focus on the following simple 2-SAT formula with only one clause

$$(v_1 \vee v_2),$$

where, WLOG, the polarity can be assumed positive for both literals (see Section A.3.2). The discontinuity of the voltage flow field clearly is in the line $v_1 = v_2$. Note that the gradient-like field

$$\left(\frac{1}{2}(1-v_2), \frac{1}{2}(1-v_1)\right)$$

is continuous everywhere, while the rigidity field is not, which is given in the upper-left and lower-right regions as

$$\frac{1}{2}(0, 1-v_2), \qquad \frac{1}{2}(1-v_1, 0),$$

respectively. The field points away from the line, meaning that the volume of a domain approaching this boundary will be expanded, with the expansion being greater the more unsatisfied the clause is. This is in fact a desired feature of the rigidity field as it attempts to expand the volume if the initial domain is in a frustrated region, which allows for a more thorough exploration of the

voltage space.

In conclusion, the flow field in the voltage space is dissipative everywhere except at certain hyperplanes, where the domain may be expanded in a manner that facilitates the finding of the fixed points.

A.11 $O(n^{\alpha})$, $\alpha \leq 1$, Scaling with Problem Size

In this Section we employ results from the (supersymmetric) topological-field theory (TFT) of dynamical systems [Ovc16] to prove that the *continuous-time* dynamics defined by Eqs. (A.17) is such that the system reaches a fixed point/solution plane (solution of the 3-SAT) in a time that scales with the 3-SAT instance size, n, as $O(n^{\alpha})$, with $\alpha \leq 1$. WLOG by "size" we mean the number of variables, n, in the instance at a fixed clause-to-variable density, $\alpha_r = m/n$ (see section A.3). When the system has reached an equilibrium point then all clauses, $C_j(\mathbf{v}_0)$, in the problem instance are strictly zero at the solution vector, $\mathbf{v}_0$.

Remark. *Note that for a continuous-time dynamics the time a physical system requires to reach an equilibrium point is strictly infinite, irrespective of the size of the problem/phase space. In practice, as it is done in numerical simulations, we say that the system has found the solution to the problem when all clauses are less then a threshold whose value does not depend on the size of the instance. As shown in proposition A.3.1, this threshold can be chosen to be as large as* $1/2$*, namely when* $\mathbf{C}(\mathbf{v}) < \frac{\mathbf{1}}{\mathbf{2}}$*, then* $\mathrm{sgn}(\mathbf{v})$ *is a solution vector. Therefore, when we discuss about the time to find the solution, we mean the shortest time for the dynamical system to cross a fixed threshold, in either the clauses or voltage variables.*

Using supersymmetric TFT it was shown that *instantons* are the *only* "low-energy" (col-

lective) dynamics of digital memcomputing machines, as those described by Eqs. (A.17) [DTO17, DO19b]. Instantons are families of classical trajectories in the phase space connecting critical points with given index (number of unstable directions) to critical points with a lower index (less number of unstable directions). The difference between indexes of the two critical points is typically 1, but it could be larger than 1. The reverse process (anti-instantons) connecting critical points of increasing index can only occur in the presence of noise and is "gapped", which means that even in the presence of noise it is exponentially suppressed compared to the instantonic process [Ovc16].

As shown in [DTO17, DO19b], the dynamics described by Eqs. (A.17) then proceed via a succession of instantonic "jumps" that "shed" unstable directions in going from a critical point to the next. In addition, as proved in Section A.9, if the dynamics admit solutions, periodic orbits and chaos cannot co-exit.

Since the instantonic trajectories are bounded (see Section A.6.2) the number of unstable directions is at most equal to the dimensionality of the phase space, $n+2m = n(1+2\alpha_r)$, and the latter grows linearly with problem size (at fixed density). Therefore, the total number of instantonic steps to reach equilibrium can only grow at most *linearly* with system size [DO19b]. The fact that the number of steps could scale *sub-linearly* with the system size is because an instanton can connect critical points that differ by more than one unstable direction. We now want to translate this result into the actual physical time to reach a solution.

The time associated with each instanton (the instanton "width") is *independent* of the size of the instance and depends only on the parameters α and β in Eqs. (A.17), the rate of change of the $\mathbf{x_s}$ and $\mathbf{x_l}$ variables, respectively. This can be seen by considering the path-integral form of the

topological action associated to Eqs. (A.17):

$$S = i\{Q,\Psi\} \equiv i\{Q, i\int_0^t d\tau\bar{\chi}(\tau)(\dot{\mathbf{x}}(\tau) - F(\mathbf{x}(\tau)))\}, \tag{A.24}$$

where the symbol $\{Q,\Psi\}$ means (summation over repeated indexes is understood)

$$\{Q,\Psi\} = \int_0^t d\tau \left(\chi^i(\tau)\frac{\delta}{\delta x^i(\tau)} + B_i(\tau)\frac{\delta}{\delta\bar{\chi}_i(\tau)}\right)\Psi, \tag{A.25}$$

with **B** the vector of momenta conjugate to the bosonic variables **x**, and the vectors χ and $\bar{\chi}$ representing pairs of Faddeev-Popov ghosts and anti-ghosts, respectively (fermionic/Grassmann variables).

The Lagrangian of the system can be read from Eq. (A.24). By taking the second derivative of this Lagrangian with respect to the memory variables we obtain the frequency of the instanton, and its inverse is its time width [Col77]. Let us call this time $T_{inst,j}$ for each instanton j. To this time we need to add the time, $T_{cr,j}$, the system spends on the initial critical point (local supersymmetric vacuum) before each instantonic jump. This time is also independent of the size of the problem and depends only on the degree of memory in the system, which is again dictated by the parameters α and β [BMTD18]. Let us call $T_{max} = \max_j(T_{cr,j} + T_{inst,j})$ the maximum time required to do an instantonic jump in the phase space, including the time the system spends on initial critical points. Again, this time does not depend on the size of the instance (size of the phase space), only on the parameters α and β. By putting all this together the maximum *physical* time, T_{phys}, required by the system to reach the solution of a given 3-SAT problem of size n and density α_r is then $T_{phys} \leq n(1+2r)T_{max}$. We have then proved

Proposition A.11.1. *Given solvable 3-SAT instances of n variables and fixed density α_r. The dynamics described by Eqs. (A.17) reach the solution of these instances in a physical time $O(n^\alpha)$, with $\alpha \leq 1$.*

Remark. *Although the physical time scales (sub-)linearly with problem size, its actual magnitude depends on T_{max} (the slope of the growth with respect to n), which in turn depends on the rate of change of the $\mathbf{x_s}$ and $\mathbf{x_l}$ variables. In addition, in the presence of physical noise, anti-instantons appear in the dynamics. Since anti-instantons are gapped (exponentially suppressed) they may increase somewhat the degree of the polynomial, but cannot transform a polynomial scalability into an exponential one.*

Note also that the (sub-)linear scalability obtained above does not necessarily apply to the numerical integration of Eqs. (A.17). The reason is that time discretization transforms continuous dynamics to an effective discrete map. For discrete maps topological supersymmetry is broken explicitly, namely the evolution operator does not commute with the Noether charge of the symmetry and the above analysis does not apply as is [Ovc16]. Of course, an efficient numerical method to integrate Eqs. (A.17) may still be found as we have shown in the numerical results of the main text. However, this numerical method cannot be strictly $O(n^{\alpha})$, with $\alpha \leq 1$, because different integration schemes introduce different numerical noise.

Chapter 5 and Appendix A, in full, have been submitted for publication. Bearden, Sean R.B.; Pei, Yan Ru.; Di Ventra, Massimiliano, 2020. The dissertation author is the primary researcher and author of the main text (Ch. 5), and Yan Ru Pei is the primary researcher and author of the supplementary information (Appx. A).

Appendix B

Sampling with Memcomputing: From a SAT Solver to a MaxSAT Solver

The content herein is my contribution to the supplementary information of Ref. [MPBD20]. The procedure is reviewed, but its application is omitted. Please see Ref. [MPBD20] for context.

The mode-assisted training method introduced in Ref. [MPBD20] requires sampling the mode of the model distribution of a given Restricted Boltzmann Machine (RBM). This task can be transformed to sampling the optimum of an equivalent weighted, mixed maximum satisfiability (MAX-2-SAT) optimization problem [MTD19]. To obtain high-quality samples for large models, we employ the memcomputing approach [DP13b, TD17, DT18], a novel computing paradigm that employs memory to both store and process information.

Memory Dynamics

Our implementation is based on the approach used in Ref. [BSD19] (Ch. 4) for the satisfiability (SAT) problem, appropriately modified for the MAX-2-SAT optimization problem. For a MAX-2-SAT with N variables, M_1 1-SAT clauses and M_2 2-SAT clauses, we have $i \in [[1,N]]$ and $m \in [[1,M_2]]$. In this case, the equations used to simulate a digital memcomputing machine

read

$$\dot{v}_i = b_i + \sum_m \left\{ W_{2,m} x_m^f x_m^s G_m^i + \rho(1 - x_m^f) R_m^i \right\} \tag{B.1}$$

$$\dot{x}_m^f = \beta(x_m^f + \varepsilon)(C_m - \frac{1}{4}), \tag{B.2}$$

$$\dot{x}_m^s = \alpha(1 + W_{2,m}) C_m. \tag{B.3}$$

The voltages, $v_i \in [-1, 1]$, are continuous representations of the N Boolean variables of the problem, y_i, with a false assignment represented as $v_i < 0$, a true assignment represented as $v_i > 0$, and $v_i = 0$ is ambiguous. Rather than thresholding the voltages to check the clause states, we use the clause function directly. A 2-SAT clause in Boolean form is comprised of two literals, $\{l_{i,m}, l_{j,m}\}$, where a literal in the m-th clause, $l_{i,m}$, is either a negated, $\bar{y}_i$, or unnegated, y_i, variable. The Boolean clause is represented as a continuous clause function,

$$C_m(v_i, v_j) = \frac{1}{2} \min[(1 - q_{m,i} v_i), (1 - q_{m,j} v_j)]. \tag{B.4}$$

The factor $q_{m,i}$ contains the information about the relation between the literal in the m-th clause, $l_{i,m}$, and its associated variable, y_i; it evaluates to $+1$ if $l_{i,m} = v_i$, and -1 if $l_{i,m} = \bar{v}_i$. The function is bounded, $C_m \in [0, 1]$, and we consider a clause to be satisfied when $C_m(v_i, v_j) < 0.5$. By thresholding the clause function we also avoid the ambiguity associated with $v_i = 0$.

Each clause has a "fast", x_m^f, and a "slow", x_m^s, memory variable that serve as indicators of the history of the state of $C_m(v_i, v_j)$. The memory is "fast" in the sense that it contains information of the *recent* history of C_m, and "slow" in the sense that it contains information on the *entire* history of C_m. Both memory variables are bounded, $x_m^f \in [0, 1]$ and $x_m^s \in [1, 10 * M_2]$. The offset $\varepsilon = 10^{-3}$ in Eq. (B.2) is used to remove spurious steady-state solutions.

The gradient-like term in Eq. (B.1) is $G_m^i = 0$ if variable y_i is not associated with any

literal in clause m. Otherwise,

$$G_m^i = q_{m,i}\frac{1}{2}(1 - q_{m,j}v_j), \tag{B.5}$$

where v_j is the value of the voltage corresponding to the other literal in the clause. The "rigidity" term in Eq. (B.1) is

$$R_m^i = \begin{cases} q_{m,i}\frac{1}{2}(1 - q_{m,i}v_i), & C_m(v_i, v_j) = \frac{1}{2}(1 - q_{m,i}v_i) \\ 0, & C_m(v_i, v_j) = \frac{1}{2}(1 - q_{m,j}v_j). \end{cases} \tag{B.6}$$

This term only influences the voltage that is closest to the satisfying assignment in the clause.

The weight of each 2-SAT clause, $W_{2,m}$, is incorporated in the dynamics of the slow memory variable and the dynamics of voltages. The weights of the 1-SAT clauses are used to bias the voltage dynamics in Eq. (B.1) as $b_i = (W_{1,i} - W_{1,\bar{i}})/2$, where $W_{1,i}$ is the weight of the 1-SAT with a literal that is equivalent to variable y_i and $W_{1,\bar{i}}$ is the weight of the 1-SAT with a literal that is the negation of variable y_i. The weight is zero if no corresponding 1-SAT exists.

The parameter values used for the simulations reported in the main text are $\alpha = 10$, $\beta = 0.1$, $\rho = 0.1$. At $t = 0$, voltages are randomly initialized with $x_m^f = 0$ and $x_m^s = 1 + W_{2,m}$. The equations are then numerically integrated with the forward Euler method using an adaptive time step, $\Delta t \in [2^{-5}, 2^{-1}]$, until a total integration time of $t = 500$ is reached. Then, we take the configuration with the lowest number of unsatisfied clauses as the sample.

Appendix B, in part, is a reprint of the material as it appear in the supplementary information of Mode-Assisted Unsupervised Learning of Restricted Boltzmann Machines in Communications Physics, 2020, Manukian, Haik; Pei, Yan Ru; Bearden, Sean R.B.; Di Ventra, Massimiliano, Nature Publishing Group, 2020. The dissertation author is a co-author of this publication.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[AJM05] Dimitris Achlioptas, Haixia Jia, and Cristopher Moore. Hiding satisfying assignments: two are better than one. *Journal of Artificial Intelligence Research*, 24:623–639, 2005.

[Bac78] John Backus. Can programming be liberated from the von neumann style?: A functional style and its algebra of programs. *Commun. ACM*, 21(8):613–641, August 1978.

[Bal16] Tomáš Balyo. Using algorithm configuration tools to generate hard random satisfiable benchmarks. *Proceedings of SAT Competition 2016: Solver and Benchmarks Descriptions*, pages 60–62, 2016.

[BC07] N. Balabanian and B. Carlson. *Digital Logic Design Principles*. John Wiley & Sons, 2007.

[BHL+02] Wolfgang Barthel, Alexander K Hartmann, Michele Leone, Federico Ricci-Tersenghi, Martin Weigt, and Riccardo Zecchina. Hiding solutions in random satisfiability problems: A statistical mechanics approach. *Physical review letters*, 88(18):188701, 2002.

[Bie17] Armin Biere. Cadical, lingeling, plingelingg, treengeling and yalsat entering the sat competition 2017. *Proceedings of SAT Competition 2017: Solver and Benchmarks Descriptions*, pages 14–15, 2017.

[BM10] Aaron R. Bradley and Zohar Manna. *The Calculus of Computation: Decision Procedures with Applications to Verification*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[BMTD18] S. R. B. Bearden, H. Manukian, F. L. Traversa, and M. Di Ventra. Instantons in self-organizing logic gates. *Physical Review Applied*, 9:034029, 2018.

[BMW00] Giulio Biroli, Remi Monasson, and Martin Weigt. A variational description of the ground state structure in random satisfiability problems. *The European Physical Journal B-Condensed Matter and Complex Systems*, 14(3):551–568, 2000.

[BMZ05] Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

[BPD20] Sean R. B. Bearden, Yan Ru Pei, and Massimiliano Di Ventra. Efficient solution of boolean satisfiability problems with memory-assisted dynamics. *?*, ?(?):?, 2020.

[BS15] Andrei A Bulatov and Evgeny S Skvortsov. Phase transition for local search on planted sat. In *International Symposium on Mathematical Foundations of Computer Science*, pages 175–186. Springer, 2015.

[BSD19] S. R. B. Bearden, F Sheldon, and M Di Ventra. Critical branching processes in digital memcomputing machines. *EPL (Europhysics Letters)*, 127(3):30005, 2019.

[BT89] Per Bak and Chao Tang. Earthquakes as a self-organized critical phenomenon. *Journal of Geophysical Research: Solid Earth*, 94(B11):15635–15637, 1989.

[CFSD17] K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta. Stochastic p-bits for invertible logic. *Phys. Rev. X*, 7:031014, Jul 2017.

[Col77] S. Coleman. *Aspects of Symmetry, Chapter 7*. Cambridge University Press, 1977.

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[Cor08] Jorge Cortes. Discontinuous dynamical systems. *IEEE Control systems magazine*, 28(3):36–73, 2008.

[Dev92] R.L. Devaney. *A First Course in Chaotic Dynamical Systems: Theory and Experiment*. Addison-Wesley, 1992.

[DO19a] M. Di Ventra and Igor V. Ovchinnikov. Digital memcomputing: from logic to dynamics to topology. *Annals of Physics*, 409:167935, 2019.

[DO19b] M. Di Ventra and Igor V. Ovchinnikov. Digital memcomputing: from logic to dynamics to topology. *Annals of Physics*, 409:167935, 2019.

[DP13a] M. Di Ventra and Y. V. Pershin. On the physical properties of memristive, memcapacitive and meminductive systems. *Nanotechnology*, 24(25), 2013.

[DP13b] M. Di Ventra and Y. V. Pershin. The parallel approach. *Nature Physics*, 9:200, 2013.

[DPC09] M. Di Ventra, Y.V. Pershin, and L.O. Chua. Circuit Elements With Memory: Memristors, Memcapacitors, and Meminductors. *Proceedings of the IEEE*, 97(10):1717–1724, Oct 2009.

[dSVBMn17] Serena di Santo, Pablo Villegas, Raffaella Burioni, and Miguel A. Muñoz. Simple unified view of branching process statistics: Random walks in balanced logarithmic potentials. *Phys. Rev. E*, 95:032115, Mar 2017.

[DT17a] M. Di Ventra and F. L. Traversa. Absence of chaos in Digital Memcomputing Machines with solutions. *Physics Letter A*, 2017.

[DT17b] M. Di Ventra and F. L. Traversa. Absence of periodic orbits in digital memcomputing machines with solutions. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27:101101, 2017.

[DT18] Massimiliano Di Ventra and Fabio L. Traversa. Perspective: Memcomputing: Leveraging memory and physics to compute efficiently. *Journal of Applied Physics*, 123(18):180901, 2018.

[DTO17] M. Di Ventra, Fabio L. Traversa, and Igor V. Ovchinnikov. Topological field theory and computing with instantons. *Ann. Phys. (Berlin)*, 529:1700123, 2017.

[ERT11] Mária Ercsey-Ravasz and Zoltán Toroczkai. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nature Physics*, 7(12):966–970, 2011.

[ES03] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *International conference on theory and applications of satisfiability testing*, pages 502–518. Springer, 2003.

[ES06] Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1–4):1–26, 2006.

[FD81] David Freedman and Persi Diaconis. On the histogram as a density estimator:l2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57(4):453–476, Dec 1981.

[FKLW03] Michael Freedman, Alexei Kitaev, Michael Larsen, and Zhenghan Wang. Topological quantum computation. *Bulletin of the American Mathematical Society*, 40(1):31–38, 2003.

[FLN07] E. Frenkel, A. Losev, and N. Nekrasov. Notes on instantons in topological field theory and beyond. *Nucl. Phys. B*, 171:215, 2007.

[Fom12] Anatolij T Fomenko. *Visual geometry and topology*. Springer Science & Business Media, 2012.

[GAE18] Aditya Grover, Tudor Achim, and Stefano Ermon. Streamlining variational inference for constraint satisfaction problems. In *Advances in Neural Information Processing Systems*, 2018.

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[GPFW99] Jun Gu, Paul W Purdom, John Franco, and Benjamin W Wah. Algorithms for the satisfiability (sat) problem. In *Handbook of Combinatorial Optimization*, pages 379–572. Springer, 1999.

[GW94] Ian P Gent and Toby Walsh. The sat phase transition. In *ECAI*, volume 94, pages 105–109. PITMAN, 1994.

[Hal10] J.K. Hale. *Asymptotic Behavior of Dissipative Systems*, volume 25 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, Rhode Island, 2nd edition, 2010.

[Heu17] Marijn J. H. Heule. Generating the uniform random benchmarks. *Proceedings of SAT Competition 2017: Solver and Benchmarks Descriptions*, page 36, 2017.

[Heu18] Marijn J. H. Heule. Generating the uniform random benchmarks. *Proceedings of SAT Competition 2018: Solver and Benchmarks Descriptions*, page 55, 2018.

[HHW96] Tad Hogg, Bernardo A. Huberman, and Colin P. Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81(1):1 – 15, 1996. Frontiers in Problem Solving: Phase Transitions and Complexity.

[HKK+00] K. Hori, S. Katz, R. Klemm, A. Pandharipande, R. Thomas, C. Vafa, R. Vakil, and E. Zaslow. *Mirror symmetry*. Clay Mathematics, 2000.

[HR04] Alexander K. Hartmann and Heiko Rieger. *New Optimization Algorithms in Physics*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2004.

[JMS07] Haixia Jia, Cristopher Moore, and Doug Strain. Generating hard satisfiable formulas by hiding solutions deceptively. *Journal of Artificial Intelligence Research*, 28:107–118, 2007.

[Kau18] Henry Kautz. Walksat version 56, 2018.

[Kit03] A Yu Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003.

[KKGvdA+18] Felipe Yaroslav Kalle Kossio, Sven Goedeke, Benjamin van den Akker, Borja Ibarz, and Raoul-Martin Memmesheimer. Growing critical: Self-organized criticality in a developing neural system. *Phys. Rev. Lett.*, 121:058301, Aug 2018.

[KMZ12] Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Reweighted belief propagation and quiet planting for random k-sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 8(3-4):149–171, 2012.

[KPP12] Robert Kozma, Robinson E. Pino, and Giovanni E. Pazienza. *Advances in Neuromorphic Memristor Science and Applications*. Springer Publishing Company, Incorporated, 2012.

[KZ09] Florent Krzakala and Lenka Zdeborová. Hiding quiet solutions in random constraint satisfaction problems. *Physical review letters*, 102(23):238701, 2009.

[LH91] E. T. Lu and R. J. Hamilton. Avalanches and the distribution of solar flares. *The Astrophysical Journal*, 380:L89–L92, October 1991.

[Mar09] Victor W Marek. *Introduction to mathematics of satisfiability*. CRC Press, 2009.

[MKHT20] Ferenc Molnár, Shubha R. Kharel, Xiaobo Sharon Hu, and Zoltán Toroczkai. Accelerating a continuous-time analog sat solver using gpus. *Computer Physics Communications*, 256:107469, 2020.

[MM09] M Mezard and A Montanari. *Information, Physics, and Computation*. Oxford University Press, 2009.

[MPBD20] Haik Manukian, Yan Ru Pei, Sean RB Bearden, and Massimiliano Di Ventra. Mode-assisted unsupervised learning of restricted boltzmann machines. *Communications Physics*, 3(1):1–8, 2020.

[MPZ02] Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

[MS08] Joao Marques-Silva. Practical applications of boolean satisfiability. In *2008 9th International Workshop on Discrete Event Systems*, pages 74–80. IEEE, 2008.

[MTD17] H. Manukian, F. L. Traversa, and M. Di Ventra. Memcomputing numerical inversion with self-organizing logic gates. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–6, 2017.

[MTD19] Haik Manukian, Fabio L Traversa, and Massimiliano Di Ventra. Accelerating deep learning with memcomputing. *Neural Networks*, 110:1–7, 2019.

[Mun14] James Munkres. *Topology*. Pearson Education, 2014.

[MZ96] Rémi Monasson and Riccardo Zecchina. Entropy of the k-satisfiability problem. *Phys. Rev. Lett.*, 76:3881–3885, May 1996.

[MZ97] Rémi Monasson and Riccardo Zecchina. Statistical mechanics of the random *k*-satisfiability model. *Phys. Rev. E*, 56:1357–1370, Aug 1997.

[MZ02] Marc Mézard and Riccardo Zecchina. Random k-satisfiability problem: From an analytic solution to an efficient algorithm. *Physical Review E*, 66(5):056126, 2002.

[MZ09] Sharad Malik and Lintao Zhang. Boolean satisfiability from theoretical hardness to practical success. *Communications of the ACM*, 52(8):76–82, 2009.

[MZK+99] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400(6740):133–137, 1999.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[Nea01] Donald A Neamen. *Electronic circuit analysis and design*, volume 2. McGraw-Hill New York, NY., 2001.

[NSS+08] Chetan Nayak, Steven H Simon, Ady Stern, Michael Freedman, and Sankar Das Sarma. Non-abelian anyons and topological quantum computation. *Reviews of Modern Physics*, 80(3):1083, 2008.

[NVS16] R. A. Nawrocki, R. M. Voyles, and S. E. Shaheen. A mini review of neuromorphic architectures and implementations. *IEEE Transactions on Electron Devices*, 63(10):3819–3829, Oct 2016.

[Ovc16] I. V. Ovchinnikov. Introduction to supersymmetric theory of stochastics. *Entropy*, 18:108, 2016.

[Par03] Giorgio Parisi. Some remarks on the survey decimation algorithm for k-satisfiability, 2003.

[Par10] Behrooz Parhami. *Computer arithmetic*, volume 20. Oxford university press, 2010.

[Pet15] Justyna Petke. *Bridging Constraint Satisfaction and Boolean Satisfiability*. Springer, 2015.

[Pru12] Gunnar Pruessner. *Self-Organised Criticality: Theory, Models and Characterisation*. Cambridge University Press, 2012.

[PSJ+17] A. Parihar, N. Shukla, M. Jerry, S. Datta, and A. Raychowdhury. Computational paradigms using oscillatory networks based on state-transition devices. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3415–3422, May 2017.

[RSR99] M Radecka, P Sobaś, and M Rekas. Ambipolar diffusion in tio2. *Solid State Ionics*, 119(1-4):55–60, 1999.

[Sau12] Timothy Sauer. *Numerical Analysis*. Pearson, 2nd edition, 2012.

[SBHF99] H.T. Siegelmann, A. Ben-Hur, and S. Fishman. Computational complexity for continuous time dynamics. *Physical Review Letters*, 83:1463, 1999.

[SD12] Alexander Stotland and Massimiliano Di Ventra. Stochastic memory: Memory enhancement due to noise. *Physical Review E*, 85(1):011116, 2012.

[SK93] Bart Selman and Henry Kautz. Domain-independent extensions to gsat: Solving large structured satisfiability problems. In *IJCAI*, volume 93, pages 290–295. Citeseer, 1993.

[SS98] T. Schäfer and E.V. Shuryak. Instantons in qcd. *Rev. Mod. Phys.*, 70:323, 1998.

[SSSW08] Dmitri B Strukov, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. The missing memristor found. *Nature*, 453(7191):80–83, 2008.

[STD19] Forrest Sheldon, Fabio L. Traversa, and Massimiliano Di Ventra. Taming a nonconvex landscape with dynamical long-range order: Memcomputing ising benchmarks. *Phys. Rev. E*, 100:053311, Nov 2019.

[Tan61] J. C. Tanner. A derivation of the Borel distribution. *Biometrika*, 48(1-2):222–224, 06 1961.

[TCSD18] F. L. Traversa, P. Cicotti, F. Sheldon, and M. Di Ventra. Evidence of exponential speed-up in the solution of hard optimization problems. *Complexity*, 2018:7982851, 2018.

[TD15] Fabio Lorenzo Traversa and Massimiliano Di Ventra. Universal memcomputing machines. *IEEE Trans. Neural Netw. Learn. Syst.*, 26(11):2702, 2015.

[TD17] Fabio Lorenzo Traversa and Massimiliano Di Ventra. Polynomial-time solution of prime factorization and np-complete problems with digital memcomputing machines. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27:023107, 2017.

[TD18] F. L. Traversa and M. Di Ventra. Memcomputing integer linear programming. *arXiv:1808.09999*, 2018.

[Tof80] Tommaso Toffoli. Reversible computing. In *International Colloquium on Automata, Languages, and Programming*, pages 632–644. Springer, 1980.

[Ver06] Ferdinand Verhulst. *Nonlinear differential equations and dynamical systems*. Springer Science & Business Media, 2006.

[Wit88] E. Witten. Topological quantum field theory. *Comms. in Math. Phys.*, 117:353–386, 1988.

[ZC92] S. Zhang and A. G. Constantinides. Lagrange programming neural networks. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(7):441–452, 1992.

[ZCO+15] H. Zhang, G. Chen, B. C. Ooi, K. Tan, and M. Zhang. In-memory big data management and processing: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1920–1948, 2015.

[ZLS95] Stefano Zapperi, Kent Bækgaard Lauritsen, and H. Eugene Stanley. Self-organized branching processes: Mean-field theory for avalanches. *Phys. Rev. Lett.*, 75:4071–4074, Nov 1995.