Hindawi
Complexity
Volume 2018, Article ID 7982851, 13 pages
https://doi.org/10.1155/2018/7982851

Research Article

Evidence of Exponential Speed-Up in the Solution of Hard Optimization Problems

Fabio L. Traversa,[1] Pietro Cicotti,[2] Forrest Sheldon,[3] and Massimiliano Di Ventra[3]

[1]*MemComputing Inc., San Diego, CA 92130, USA*
[2]*San Diego Supercomputer Center, La Jolla, CA 92093, USA*
[3]*Department of Physics, University of California, La Jolla, San Diego, CA 92093, USA*

Correspondence should be addressed to Massimiliano Di Ventra; diventra@physics.ucsd.edu

Received 17 April 2018; Accepted 29 May 2018; Published 3 July 2018

Academic Editor: Viet-Thanh Pham

Optimization problems pervade essentially every scientific discipline and industry. A common form requires identifying a solution satisfying the maximum number among a set of many conflicting constraints. Often, these problems are particularly difficult to solve, requiring resources that grow exponentially with the size of the problem. Over the past decades, research has focused on developing heuristic approaches that attempt to find an approximation to the solution. However, despite numerous research efforts, in many cases even approximations to the optimal solution are hard to find, as the computational time for further refining a candidate solution also grows exponentially with input size. In this paper, we show a *noncombinatorial* approach to hard optimization problems that achieves an *exponential speed-up* and finds better approximations than the current state of the art. First, we map the optimization problem into a Boolean circuit made of specially designed, *self-organizing* logic gates, which can be built with (nonquantum) electronic elements with memory. The equilibrium points of the circuit represent the approximation to the problem at hand. Then, we solve its associated *nonlinear* ordinary differential equations numerically, towards the equilibrium points. We demonstrate this exponential gain by comparing a sequential MATLAB implementation of our solver with the winners of the 2016 Max-SAT competition on a variety of hard optimization instances. We show empirical evidence that our solver scales *linearly* with the size of the problem, both in time and memory, and argue that this property derives from the *collective* behavior of the simulated physical circuit. Our approach can be applied to other types of optimization problems, and the results presented here have far-reaching consequences in many fields.

1. Introduction

In real-life applications, it is common to encounter problems where one needs to find the best solution within a vast set of possible solutions. These *optimization problems* are routinely faced in many commercial segments, including transportation, goods delivery, software packages or hardware upgrades, network traffic and congestion management, and circuit design, to name just a few [1, 2]. Many of these problems can be easily mapped into *combinatorial optimization problems*, namely, they can be written as Boolean formulas with many constraints (clauses) among different variables (either negated or not, i.e., literals) with the constraints themselves related by some logical proposition [1].

It is typical to write the Boolean formulas as *conjunctions* (the logical ANDs, also represented by the symbol $\wedge$) of *disjunctions* (the logical ORs, represented by the symbol $\vee$), in the so called *conjunctive normal form* (CNF). The CNF representation is universal in that any Boolean formula can be written in this form [3].

A simple example of a CNF formula $\phi(x)$ is

$$\phi(x) = (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_4) \wedge (x_1 \vee x_2 \vee \neg x_4), \quad (1)$$

in which we have four variables, x_j, with $j = 1, 2, 3, 4$, five clauses, and fourteen literals (the symbol $\neg$ indicates

negation). The problem is then to find an assignment satisfying the maximum number of clauses, that is, in which as many clauses as possible have at least one literal that is true. Such a clause is then said to be satisfied, otherwise it is unsatisfied [3], and the problem itself is known as Max-SAT (maximum satisfiability).

A Max-SAT problem whose CNF representation has exactly k literals $(k \geq 2)$ per clause is called Max-EkSAT. Max-EkSAT is a ubiquitous optimization problem with widespread industrial applications. We will focus on its solution as a test bed in the main text and refer the reader to the appendix where we have applied our approach to a wide range of optimization problems, including weighted Max-SAT, [4] for its application to machine learning and [5] for the solution of the worst cases of a satisfiable problem known as the subset sum.

Max-EkSAT lies in the NP-hard class, meaning that any problem in NP can be reduced to it in polynomial time [1]. More informally, we expect that worst case instances will require resources which grow (at least) exponentially in the input size to solve, and additionally, problems in this class generally also require exponential resources in order to check a proposed solution. Due to this, complete algorithms that attempt to solve Max-EkSAT instances quickly become infeasible for large problems. Much research has instead focused on incomplete solvers that perform a stochastic local search, by generating an initial assignment and iteratively improving upon it. This approach has proven effective at approximating and sometimes solving large instances of SAT and other problems. For instance, in recent Max-SAT competitions [6], incomplete solvers outpace complete solvers by two orders of magnitude on random and crafted benchmarks. However, they too suffer from the same exponential time dependence as complete solvers for sufficiently large or hard instances [7–9].

It has further been shown, using probabilistically checkable proofs [10], that many classes of combinatorial optimization problems (including the Max-EkSAT) have an *inapproximability gap*. This means that no algorithm can overcome, in polynomial time, a fraction of the optimal solution, unless NP = P [10, 11]. In other words, for heuristics to improve on their approximation beyond this limit would require exponentially increasing time. For example, for the Max-E3SAT, it has been proved that if NP ≠ P, then there is no algorithm that can give an approximation better than 7/8 of the optimal number of satisfied clauses [11].

Despite these difficulties, it is often necessary to solve or approximate optimization problems such as these as quickly as possible, and the quality of the approximation can have direct outcomes on the cost to businesses, the speed of our internet connections, or the efficiency of our shipping, to name a few important cases. In what follows, we outline a novel approach to generating approximations to Max-EkSAT and demonstrate its efficacy on a variety of instances both generated to provide the worst cases within the inapproximability gap and drawn from Max-SAT competitions [6].

2. The Memcomputing Approach

In this work, we consider a radically different *noncombinatorial* approach to hard optimization problems. Our approach is based on the *simulation* of *digital memcomputing machines* (DMMs) [5, 12, 13]. A brief introduction of these machines is provided in the appendix. The reader interested in a more in-depth discussion is urged to look at the extensive papers [5, 12]. The practical realization of DMMs can be accomplished using standard circuit elements and those with memory (time nonlocality, hence the name "memcomputing" [14]).

Time nonlocality allows us to build logic gates that *self-organize* into their logical proposition, *irrespective* of whether the signal comes from the traditional input or output [12]. We call them *self-organizing logic gates* (SOLGs), and circuits built out of them, *self-organizing logic circuits* (SOLCs). Our approach then follows these steps.

(1) We first construct the Boolean circuit that represents the problem at hand (e.g., the Max-EkSAT of Figure 1).

(2) We replace the traditional (unidirectional) Boolean gates of this Boolean circuit with SOLGs.

(3) We feed the appropriate terminals with the required output of the problem (e.g., the logical 1 if we are interested in checking its satisfiability).

(4) Finally, the electronic circuit built out of these SOLGs can be described by *nonlinear* ordinary differential equations, which can be solved to find the equilibrium (steady-state) points. These equilibria represent the approximation to the optimization problem [12].

The procedure of how we transform a combinatorial optimization problem into an electronic circuit as well as a sketch of its numerical solution is discussed further in the appendix (see also [12]). The important point to note is that SOLGs and SOLCs manifest *long-range order* due to the presence of instantons [15]. Instantons connect topologically inequivalent critical points in the phase space, hence generating *nonlocality* in the system. This translates into a *collective* dynamical behavior that allows gates at an *arbitrary* distance to correlate very efficiently so that, when a terminal of one gate needs to change its truth value to satisfy that gate's logical proposition, a terminal at any other gate may provide the correct truth assignment while satisfying its own logical proposition [5]. As we will explain later, this is the key feature that allows these memcomputing machines to solve complex problems efficiently, without the need to explore a vast space of possibilities, as standard combinatorial approaches would do.

3. Results and Discussion

This radical change of perspective manifests its power already in comparing simulations of DMMs with those performed by the winners of the 2016 Max-SAT competition

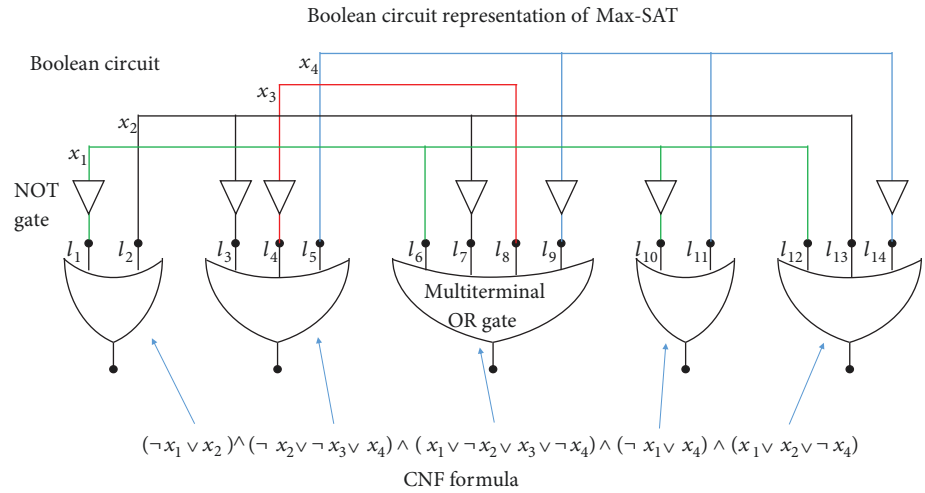


Figure 1: Example of the mapping between a Boolean satisfiability formula in conjunctive normal form and a Boolean circuit made of multiterminal OR and NOT gates. Each clause of the SAT formula is mapped into an OR with as many terminals as the literals in the clause (the satisfiability of this multiterminal OR requires that at least one terminal has a truth value of 1). The global optimum of the SAT formula, that is, the maximum number of satisfied clauses, corresponds to the maximum number of OR gates with output one. This Boolean circuit is then transformed into a self-organizing logic circuit by substituting each standard Boolean gate with a self-organizing logic gate [12], and each OR output is fed with a DC voltage generator representing the logic value of 1.

[6] on the competition benchmarks. When run on similar hardware, our solver, which we named Falcon [12, 16, 17], performs orders of magnitude faster than the winners in the incomplete track of the competition, and in some cases it finds the solution when the best solvers did not.

Since a direct comparison is difficult across hardware and implementations (our solver is written in MATLAB which is notoriously inefficient compared with the compiled languages of the competition solvers). Nevertheless, these tests already provide strong indication of the advantages of our approach using digital memcomputing machines over traditional combinatorial optimization.

However, in order to form a direct comparison and more clearly show the *exponential speed-up* of our approach, we have crafted three Max-SAT problems with increasing levels of difficulty. We then compared our memcomputing solver against two of the best solvers of the 2016 Max-SAT competition (CCLS [18] and DeciLS [19]—a new version of CnC-LS—kindly provided by their developers) which are specifically designed to solve these types of problems, but employing very different solution strategies.

Random 3-SAT instances may be generated by selecting 3 variables out of n, joining them in a 3-SAT clause where each is randomly negated and then repeating this for the desired number of clauses M. These instances are known to undergo a SAT/UNSAT transition when the ratio of clauses to variables, $M/n = \rho$ (hereafter the "density"), crosses the critical value $\rho_c \approx 4.3$ [20, 21]. Exponential time is required to demonstrate that an instance is UNSAT [22] and thus must also be required to solve the corresponding Max-SAT, offering a simple way to generate benchmarks.

However, the difficulty of computing approximations for these instances varies widely. This can be partially attributed to the fluctuations in variable occurrences and their negations [23] leading to "fields" which point towards the optima. More balanced instances may be produced by starting with a Random-XORSAT instance (also called hyperSAT [24]), that is, a set of Boolean formulas defined by the XOR of Boolean variables (the XOR symbol is $\oplus$) and converting it to a Max-SAT instance.

Each XORSAT clause may be converted to a block of four SAT clauses, for example,

$$x \oplus y \oplus z = 1 \rightarrow (x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z), \tag{2}$$

in which a variable and its negation appear symmetrically. The special structure of XORSAT gives rise to a global algorithm when the instance is satisfiable, allowing for a solution in polynomial time using Gaussian elimination [24]. However, when unsatisfiable, occurring for $\rho > 4 \cdot 0.918 \approx 3.7$, this same structure makes these problems very difficult for local search solvers [22, 25]. In addition, the choice of instances out of XORSAT clauses makes them particularly difficult also for algorithms based on message passing [26].

A basic understanding of this difficulty can be obtained by considering that changing a variable assignment affects positively (namely, contributes a true literal to) the same number of clauses as those affected negatively (where the literal is false), because of the balanced occurences of the variables. Therefore, for any combinatorial approach, when a certain amount of satisfied clauses is reached, any further improvement requires many *simultaneous* variable flips, which is a *nonlocal* type of assignment. In other words, the distance between two assignments at successive approximations becomes of the same order of the input length $|x|$. This means that going from an assignment x to a better one y, if they have a distance $d(x, y) = \sum_j (x_j - y_j)^2 = O(|x|)$, would

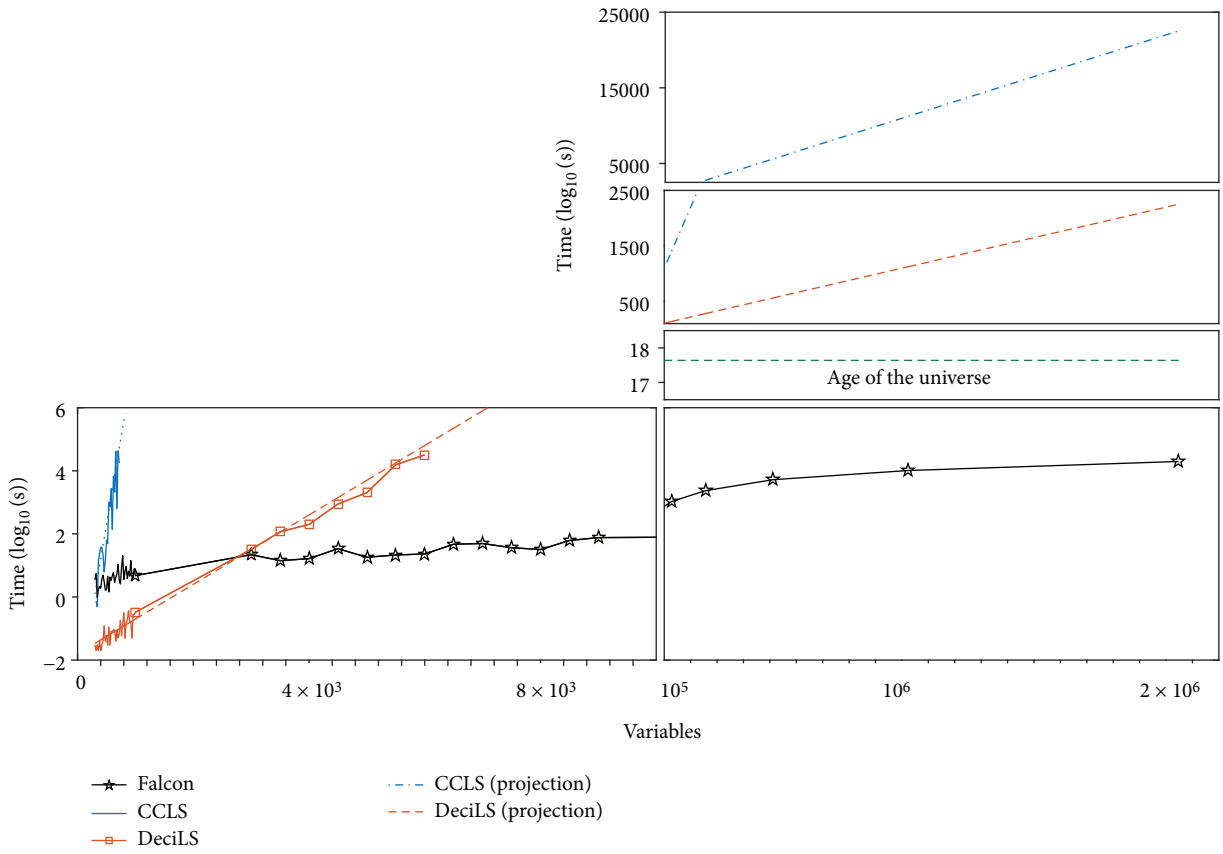


Figure 2: Simulation time comparison between incomplete solvers CCLS and DeciLS against our solver, Falcon, for the balanced and constrained delta-Max-E3SAT. A threshold of 1.5% of unsatisfiable clauses has been set. We have then tested how long CCLS, DeciLS, and our solver Falcon take to overcome this limit with increasing number of variables. All calculations have been performed on a single thread of an Intel Xeon E5-2680 v3 with 128 Gb DRAM shared on 24 threads. The local solvers require an exponentially increasing time to reach that limit already visible at a few hundred variables for the CCLS and a few thousands for the DeciLS. Our solver has been tested up to 2×10^6 variables and required order of 10^4 seconds for that maximum number of variables. We show also the estimate of time that would have been required these local solvers to run up to 2×10^6 variables. The estimated time (dashed and dashed-dotted lines) has been calculated using a linear regression of the $\log_{10}(\text{time})$ versus the number of variables.

require checking $O(2^{d(x,y)})$ variable flips, which is the number of configurations that is exponential with respect to the distance $d(x, y)$ (the actual calculation requires the enumeration of all possible flips of $1, 2, \ldots, d(x, y)$ literals because the distance $d(x, y)$ is not known a priori. Hence, the actual number of flips is $\sum_{k=0}^{d(x,y)} \binom{|x|}{k} \geq 2^{d(x,y)}$).

While more difficult, these instances also display some variation in resolution time. In order to obtain instances of more predictable difficulty, we impose a further constraint requiring all variables to appear the same number of times (or as near as possible while remaining consistent with the number of clauses $M = \rho N$), that is, the variable occurrences are distributed as a δ-function. This variant is harder than the previous one because of the additional balance induced by the variable distribution, and our results indicate that they display much lower variability in their difficulty.

In the following, we will call "random-Max-E3SAT" a Max-E3SAT completely generated at random. This will be used as an "easy" problem to test the performance of all solvers. We refer to "hyper-Max-E3SAT" as the Max-E3SAT generated from a random Max-E3XOR and finally to "delta-Max-E3SAT" as a problem generated by the Max-E3XOR with δ-function distribution of variables.

As it is evident in Figure 2, while the balanced structure of Max-XORSAT poses a challenge to local search algorithms, our memcomputing solver easily overcomes these limits because, due to the collective (instantonic) behavior of the circuit, the dynamics evolve towards deep minima very close to the global optimum (see also the appendix). The reason is that, as already anticipated, the collective state of the machine allows simultaneous, *nonlocal* change of literals belonging to gates arbitrarily far from each other [15]. This change is consistent with the physics and the topology of the memcomputing circuit that naturally drive the system towards the maximum number of satisfied SOLGs, without recourse to any combinatorial selection scheme.

The optimum for all problems can be estimated using an ensemble of small instances for which it is easier to find a fairly good approximation. For example, instances of about 300 variables and density (clauses/variables) of $\rho = 5$ provide a good indication of the global optimum in terms of percentage of unsatisfied clauses. We found that for the random-

Max-E3SAT, the optimum is expected at about 0.4% of unsatisfied clauses, while for both the hyper- and delta-Max-E3SAT, this value is about 1.3%. The difference between these values is not surprising. As mentioned previously, it is well known that for the latter two problems the transition from satisfiable to unsatisfiable is around a density of $\rho \approx 3.7$, while for random-Max-E3SAT, it is around $\rho \approx 4.3$. We have then chosen the same density of $\rho = 5$ for the random-, hyper-, and delta-Max-E3SAT.

In order to prove the superior efficiency of our noncombinatorial approach for this class of hard problems, we have evaluated their scaling properties up to 2×10^6 variables (while keeping the density constant). We recall that the simulations of DMMs have been done using a MATLAB code, while CCLS and DeciLS are compiled codes. Therefore, the level of optimization is expected to be higher in the compiled codes, making a direct performance comparison harder, although for large problem sizes, our solver has much better performance compared to CCLS and DeciLS. Nevertheless, we are more interested in the scaling of the approximation time. Specifically, for hard cases where incomplete solvers diverge exponentially in time, our solver diverges *linearly*. This is the most important test and the central result of our paper. It is shown in Figure 2.

The hard inapproximability limit and its exponential nature for both the combinatorial heuristics CCLS and DeciLS is clearly visible in Figure 2, where we have set a threshold of 1.5% of unsatisfiable clauses for the delta-Max-E3SAT. We have then tested how long CCLS, DeciLS, and our solver Falcon take to overcome this limit with increasing number of clauses. All calculations have been done on a single core of an Intel Xeon E5-2680 v3.

The exponential blowup of CCLS and DeciLS is already evident for small instances of the problem, while our noncombinatorial approach performs *linearly*, in both time and memory, for *any* number of variables we have tested so far. In fact, we have tested our solver up to 2×10^6 variables, requiring $\sim 10^4$ seconds to reach the target 1.5% threshold. The heuristic solvers, if they could run up to the same number of variables, would require, in the best case, about $\sim 10^{2500}$ seconds, which is $\sim 10^{2480}$ times the estimated age of the universe.

To better highlight the *linear* scaling of our solver, we compare it in Figure 3 with CCLS (qualitatively, all other incomplete solvers should perform similarly). Each plot of Figure 3 displays the percentage of unsatisfied clauses versus time, normalized with respect to the number of variables n. Clearly, linear scaling for these hard problems is a very desirable feature and very difficult to achieve with combinatorial approaches. However, the reason for such linear scaling is subtle.

Regarding *memory*, since we simulate (integrate) differential equations in time, and the circuit scales linearly with the number of literals, the linear scaling in memory requirements of our simulations is easy to understand (see also the appendix). On the other hand, linear scaling in simulation *time* implies *constant* scaling, namely, *independent* of the problem size, when we look at the "machine time," which is the number of (differential equation discretized time) steps for the simulation to reach equilibrium. The reason for this unexpected machine time constant scaling can be found again in the long-range order of the dynamics of the system [15] (see also the appendix). As we have shown analytically in [15] using topological field theory, this long-range order leads to nondecreasing spatial (and temporal) correlations in memcomputing machines. In fact, Figure 3 clearly shows that self-organizing logic circuits relax close to the predicted global minimum, while the CCLS does so only for the ("easy") random-Max-E3SAT. This is further illustrated in Figure 4 of the appendix for random-, hyper-, and delta-Max-E3SAT.

4. Conclusions

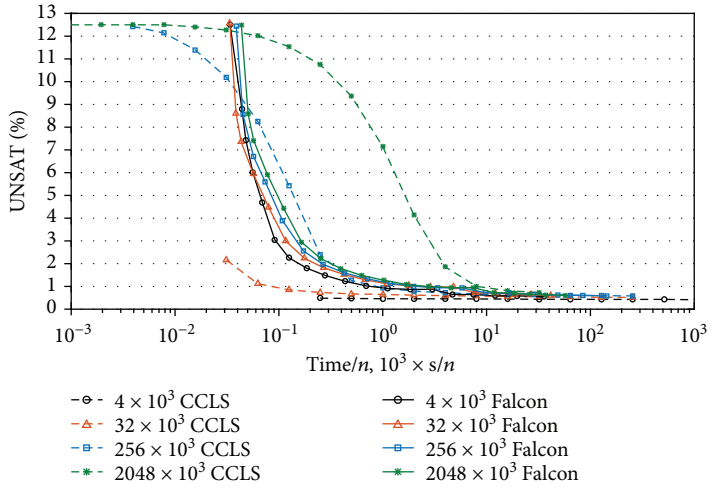
In conclusion, we have shown empirical evidence that a *noncombinatorial* approach—based on the simulation of digital memcomputing machines—to the solution of hard combinatorial optimization problems outperforms *exponentially* heuristics specifically designed to solve such problems. In particular, with our approach, we were able to find far better approximations to hard instances with millions of variables in a few hours on a single core, with *linear scaling* both in time and memory of the processor. For the same sizes, winners of the 2016 Max-SAT competition would require several orders of magnitude more than the age of the universe to find the same approximations. Of course, these numerical results are not intended to prove that there are polynomial solutions to NP-hard problems. Rather, they show that physics-inspired approaches can help tremendously in solving some of the most complex problems faced in academia and industry. We thus hope that this work will motivate further research along these lines.

Appendix

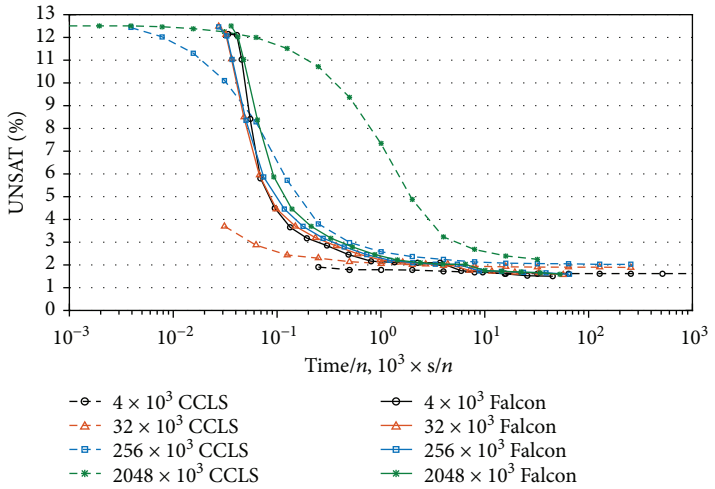
Methods

The noncombinatorial approach we discuss here is based on the concept of universal memcomputing machines (UMMs) [13] introduced by two of us (Fabio L. Traversa and Massimiliano Di Ventra). UMMs are a class of computing machines composed of interconnected memory units. The *topology* of such network is chosen to solve the specific problem at hand. UMMs use the *collective* state of the interconnected memory units to perform computation [12, 27], so they can take advantage of long-range correlations that can significantly boost the efficiency of the computation [12, 15]. If the input and output of UMMs can be mapped into strings of integers, belonging to a limited subset of $\mathbf{N}$, we obtain the digital (hence scalable) version of UMMs (DMMs) [12]. In particular, we consider DMMs whose input and output can be mapped into $\mathbf{Z}_2$.

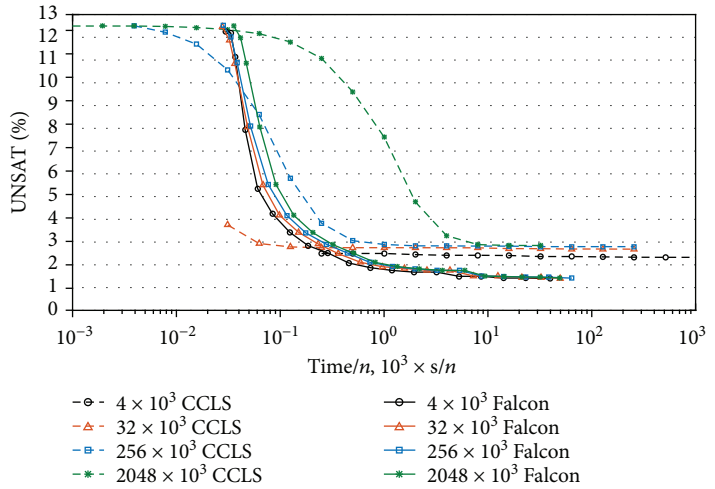
A possible, practical realization of DMMs is self-organizing logic circuits (SOLCs) composed of SOLGs [12]. SOLGs are logic gates that can accept inputs from *any* terminals and self-organize their internal state to satisfy their logic relations. For example, a self-organizing OR (SO-OR) is



(a)

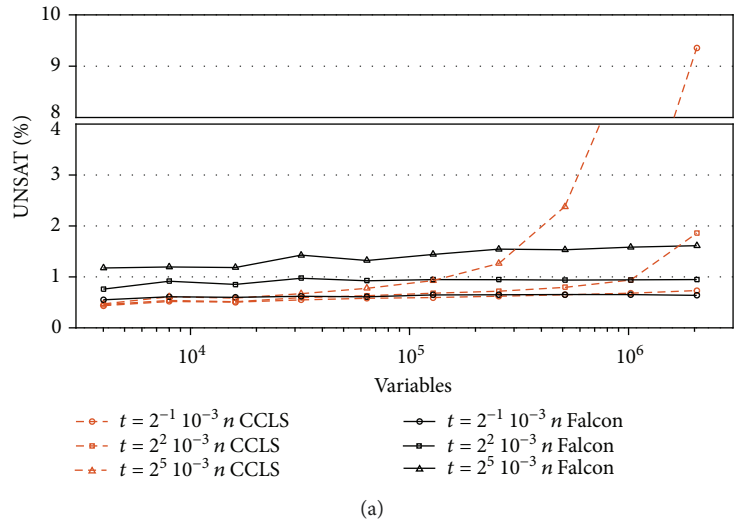


(b)

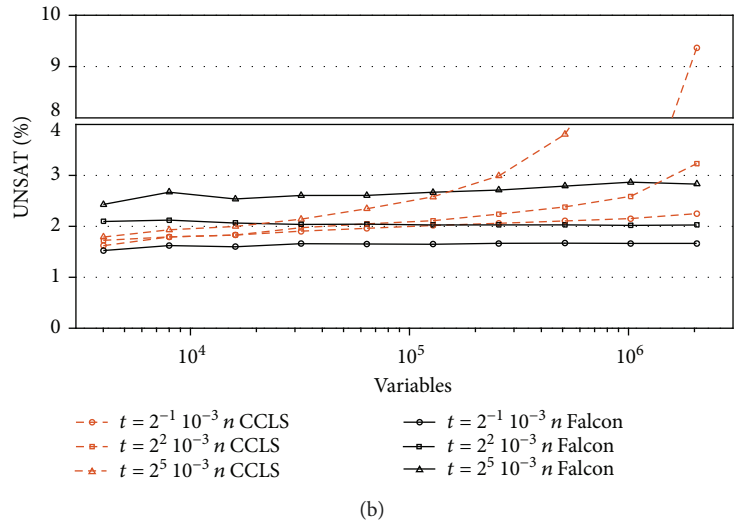


(c)

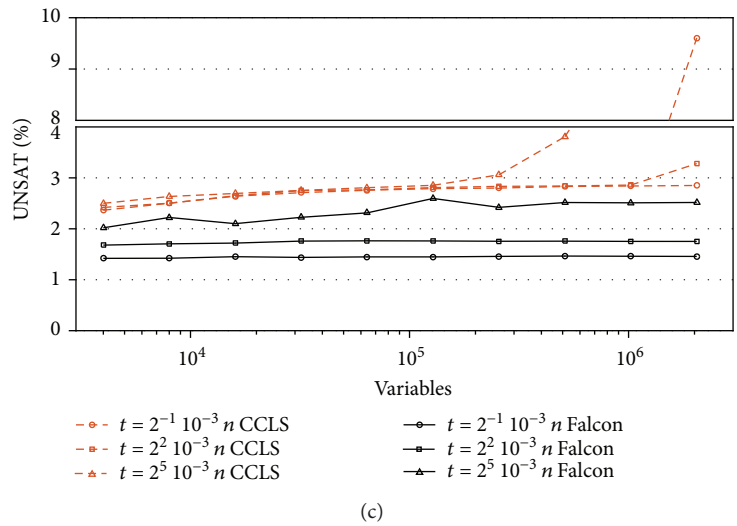
Figure 3: Comparison between the incomplete solver CCLS versus our noncombinatorial solver, Falcon, for (a) random-Max-E3SAT, (b) hyper-Max-E3SAT, and (c) delta-Max-E3SAT. In these plots, the percentage of unsatisfied clauses versus the time normalized with respect to the number of variables is shown to highlight the linear scaling of our solver. All calculations have been performed on a single thread of an Intel Xeon E5-2680 v3 with 128 Gb DRAM shared on 24 threads.



(a)



(b)



(c)

Figure 4: Comparison between the CCLS solver versus our solver, Falcon, for (a) random-Max-E3SAT, (b) hyper-Max-E3SAT, and (c) delta-Max-E3SAT. In these plots, the percentage of unsatisfied clauses versus the number of variables is shown. Different curves are for different simulation time-outs (in seconds) following the relation $t_{\text{out}} = kn$ with $n = |x|$ and k an integer given in the legend. All calculations have been performed on a single thread of an Intel Xeon E5-2680 v3 with 128 Gb DRAM shared on 24 threads.

a 3-terminal gate whose internal machinery drives the terminal states to satisfy the relation $x_0 = x_1 \vee x_2$, where x_0 is the state of the conventional output terminal, and x_1 and x_2 are the states of the conventional input terminals. Therefore, unlike conventional logic gates, the SO-OR can be fed also at the output terminal. If we set x_0 to some state, the SO-OR then will self-organize to give logically consistent states x_1 and x_2.

We can use SOLCs to solve combinatorial problems by expressing them in Boolean format and then mapping the latter onto logic circuits. As a relevant example for this work, we can take the Max-SAT problem written in CNF. When we transform the SAT into a Boolean circuit, we have multiterminal OR gates connected together in order to represent a logic formula (see Figure 1 of the main text). Hence, we can substitute conventional logic gates by SOLGs and set all output of the SO-ORs to logical 1. We now let the SOLC to self-organize to satisfy the largest number of SO-ORs.

We have previously shown [12] that SOLCs can be realized via standard (nonquantum) electronic components (we employ the realization described in [12], just slightly modified to deal with CNF formulas).

One of the key components of SOLGs is the *dynamic correction module* we have designed to correct the inconsistent logic gate configurations. While the design and details of this component can be found in [12], we recall here its working principle. The error correction module dynamically reads the voltages at the terminals of the gate and injects a large current when the gate is in an inconsistent configuration, a small current otherwise.

The nonquantum electronic nature of SOLCs can be fully described by a system of *nonlinear* ordinary differential equations of the type

$$\dot{x}(t) = \mathbf{F}(x(t)), \tag{3}$$

where $x = \{v_j, x_i\} \in X$ (X is the phase space) is the collection of voltages, v_j, at the terminals and the internal state variables, x_i, of the electronic elements with memory; $\mathbf{F}$ is a system of nonlinear ordinary differential equations, representing the flow vector field [12]. We can then efficiently simulate them by numerical integration. Therefore, SOLCs are nothing other than dynamical systems. In this case, a solution of the problem we want to solve (e.g., the Max-SAT) employing a DMM is mapped into an equilibrium point of the dynamical system. The system is engineered in such a way that, starting from *any* initial condition (generally chosen at random), it evolves to converge into an equilibrium.

We have discussed in [12] (see also [5]) the relevant properties that the dynamical systems representing DMMs should have to behave in this way. Among them, an important feature, fundamental to guarantee the convergence, is that they are *point dissipative* [28]. This implies that the dynamical system has bounded orbits (no divergences), and it is endowed with an asymptotically stable global attractor, that is, a compact set in the phase space that attracts any other point. This feature has also allowed us to prove that no chaotic behavior can emerge if equilibrium points are present [29], as well as absence of periodic orbits [30]. Finally, the point dissipative property guarantees convergence to equilibrium *irrespective* of the initial conditions.

We can finally summarize the power of these machines with the following hierarchical picture. DMMs use the *topology* of the internal connectivity of its elements to represent the problem to solve (this is called *information overhead* in [12]). Then, the collective state of the machine can manipulate all inputs, outputs, and connecting variables in a massively parallel fashion (*intrinsic parallelism* [12]).

In addition, the nonlinearity of the dynamical system equations induces a transient instantonic phase with *long-range order*, both in space and time [15]. This long-range order allows the system to converge exponentially fast to the equilibrium points that are associated to the approximations of optimization problems, by exploring a subspace (that scales at most polynomially with input size) of the phase space. This subspace is considerably smaller than the entire phase space itself [15].

In fact, as briefly discussed in the main text, the particular realization of DMMs we have presented in this work (similar to the ones in [12]) supports infinite-range correlations in the infinite input size limit, as shown in [15]. This enables an ideal scale-free behavior (namely, one where the correlations *do not* decay) of the SOLC. This was derived analytically using topological field theory in [15] and can also be supported numerically from Figure 5 as follows.

In order to simulate the system, we have employed a time step size-controlled forward-integration scheme for the differential equations that describe it [31]. Since the number of variables of the problem grows linearly with the input size because the number of gates grows only linearly, each time step to be simulated requires only a linear number of floating point operations and a memory linearly growing with input size. Then, the simulation time is just a linear function of the machine time. In Figure 5, it is reported the same as in Figure 3 of the main text but with the SOLC time (not normalized) on the x-axis. It is evident that the relaxation of the system is independent of the input size (ideal scale-free scaling). This is a very interesting and rare result for an extensive interconnected system. All these ingredients are necessary for the correct, efficient operation of a DMM.

The approximations to an optimization problem found by DMMs are very close to the global minimum of the problem, and this is guaranteed by the topology of the connectivity. This is clearly demonstrated in Figure 4 where the unsatisfied clauses are plotted versus variables for different simulation times, scaled linearly by the number of variables. While for the random-Max-E3SAT, both our solver and the CCLS approach the 0.4% minimum; in the case of the hyper-Max-E3SAT, CCLS reaches a hard *inapproximability limit* of about 2% for large instances. As expected, the delta-Max-E3SAT, instead, is a much worse case, and the inapproximability limit for CCLS is at about 3%.

In contrast, our noncombinatorial approach directly reaches the global minimum in all cases. Interestingly, our solver shows slightly better performances for the delta-Max-E3SAT (the most difficult of the three cases) as can be seen by taking a closer look at Figure 4.

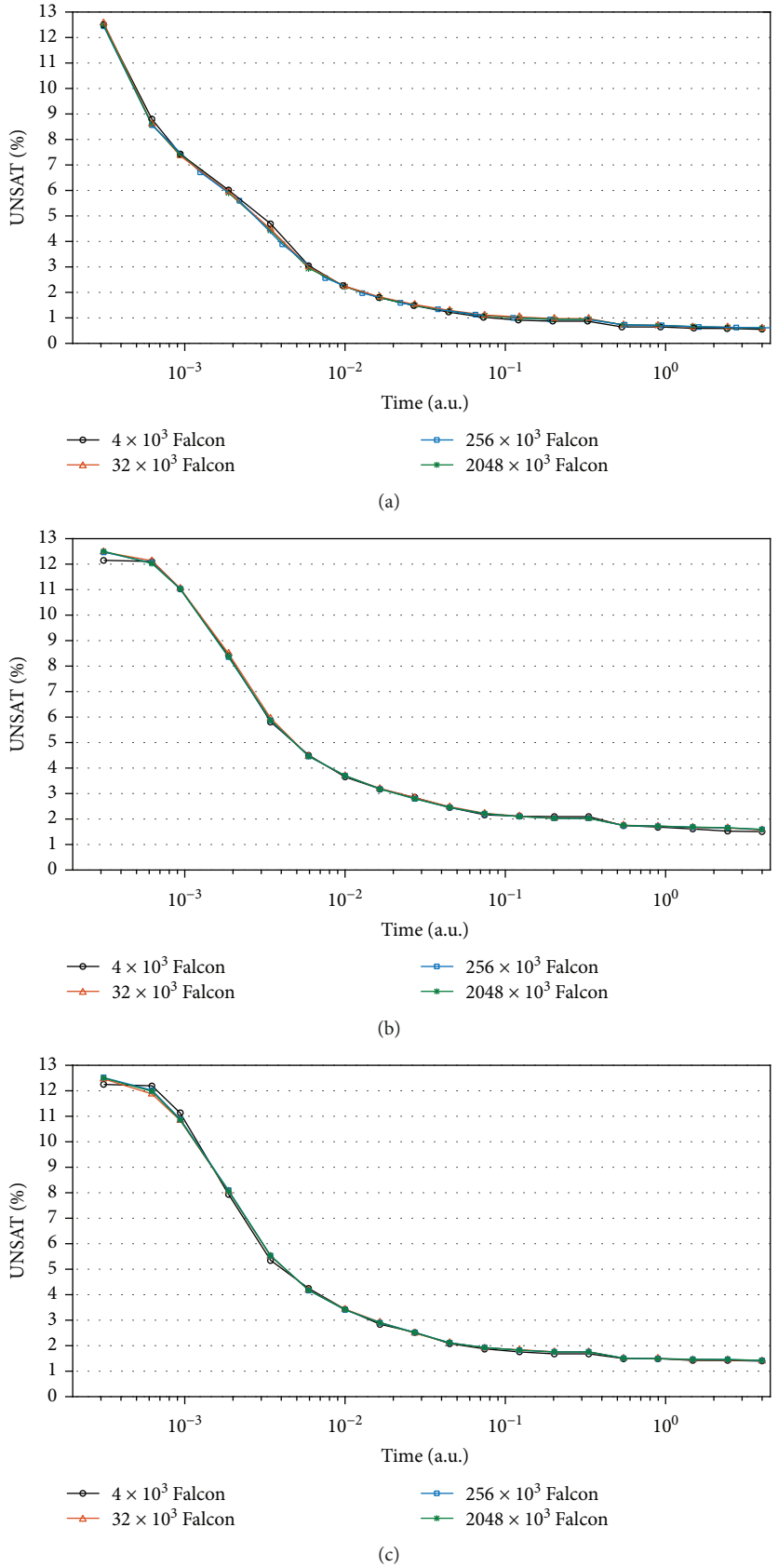


Figure 5: Percentage of unsatisfied clauses versus the machine time (i.e., simulated time steps) is shown to highlight the linear scaling of our solver, Falcon, for (a) random-Max-E3SAT, (b) hyper-Max-E3SAT, and (c) delta-Max-E3SAT. All calculations have been performed on a single thread of an Intel Xeon E5-2680 v3 with 128 Gb DRAM shared on 24 threads.

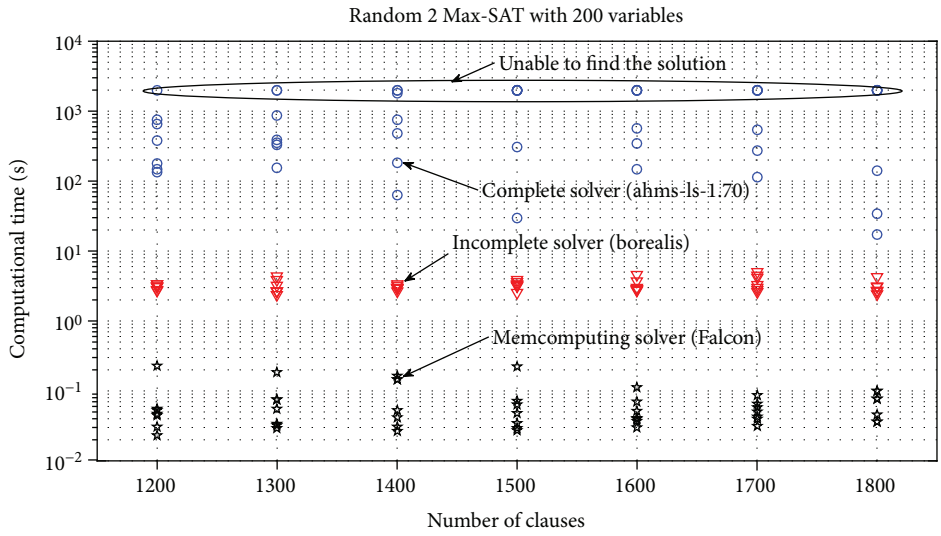


Figure 6: Results from the 2016 Max-SAT competition for the random Max-2SAT problem compared with our memcomputing solver, Falcon.

A Brief Survey on Max-SAT Solvers

As mentioned in the main text, there are two main (combinatorial) approaches to solve or approximate the Max-SAT problem. The first is based on the exhaustive exploration of the solution space and leads to the so-called "complete" solvers [7, 22]. The complete solvers use algorithms typically based on the branch-and-bound approach [1, 19] in which a greedy bound is first put on the optimum and then this is used to prune the resulting search tree. Despite this pruning, they still scale exponentially with input size $|x|$ because they exhaustively search a space $\mathbf{Z}_2^{O(|x|)} = \{0,1\}^{O(|x|)}$. However, when the computation is finished, complete solvers are guaranteed to have found the global optimum of the Max-SAT.

Incomplete solvers [7, 8], in comparison, cannot guarantee the optimality of their solution as they do not explore the entire solution space. Instead, they proceed by generating an initial assignment and iteratively improving upon it. This trade-off allows them to find solutions, when they do, much more quickly than complete algorithms. In the most recent Max-SAT competition [6], incomplete track solvers found solutions of two orders of magnitude faster than complete track solvers in random and crafted benchmarks.

The quintessential incomplete solver is WalkSAT [8] which proceeds through a stochastic local search. After an initial assignment is generated, an unsatisfied clause is selected and one variable from the clause has its assignment flipped. This will leave this clause satisfied but may alter the state of other clauses in which the variable occurs. The procedure is continued for a specified number of steps or until a solution is found. Most current local search solvers work similarly with various heuristics to select the next variable flip and utilize restarts and/or noise and a host of other features.

We compared our solver, Falcon, with two of the best solvers from the 2016 Max-SAT competition, CCLS [18] and DeciLS [19]. CCLS won the crafted track for unweighted Max-SAT and performed near the top of the random track. It performs a local search (LS) with configuration checking (CC), and the binary provided took no tuning parameters. Local search solvers will often retrace flips many times leading to an inefficient search. Configuration checking keeps track of when neighboring variables have been flipped and only allows a variable to be flipped again when at least one of its neighbors has changed its assignment. DeciLS is an updated version of CnC-LS which won the industrial track for unweighted Max-SAT and combines a unit propagation based decimation (Deci) and local search (LS) with restarts. An assignment is first generated through unit propagation-based decimation [22] in which conflicts are allowed, and the result is given to a local search for a specified number of steps. The process is then restarted, and the best result of the previous search is used to guide the subsequent decimation and resolve conflicts. This allows the solver to explore very different reasoning chains and areas of the solution space. We used the parameters values recommended in [19] for good performance across a range of instances, and subsequent tuning has indicated that the results are insensitive to changes in this range.

Weighted Partial Max-SAT

In order to more efficiently map a large number of maximization problems into Max-SAT, it is sometimes useful to consider a variant: *weighted* partial Max-SAT [1, 19]. Weighted partial Max-SAT is a version of Max-SAT for which a subset of clauses *must* be satisfied ("hard" clauses), while the remaining clauses ("soft" clauses) may be weighted, and the sum of the weights of satisfied clauses must be maximized. The Max-SAT is a particular case of the weighted partial Max-SAT in which all clauses are soft and have the same weight.

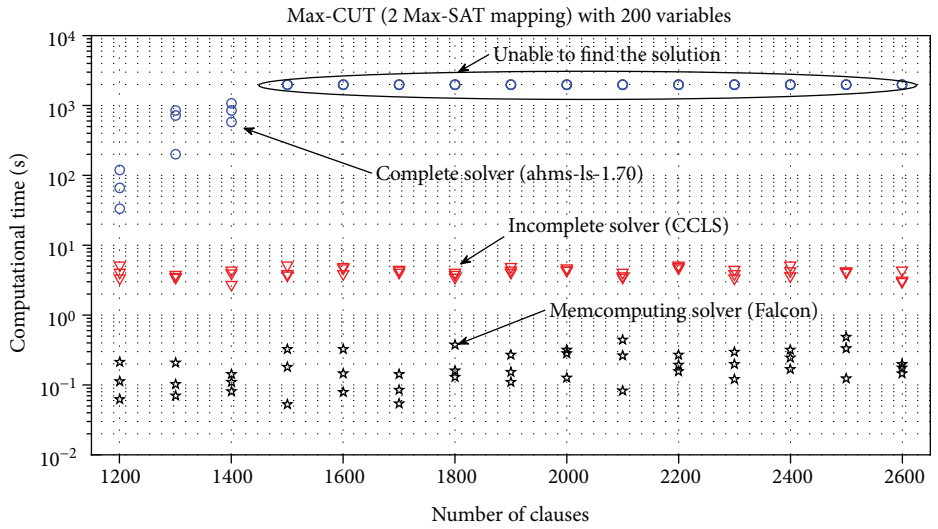


Figure 7: Results from the 2016 Max-SAT competition for the Max-CUT problem compared with our memcomputing solver, Falcon.

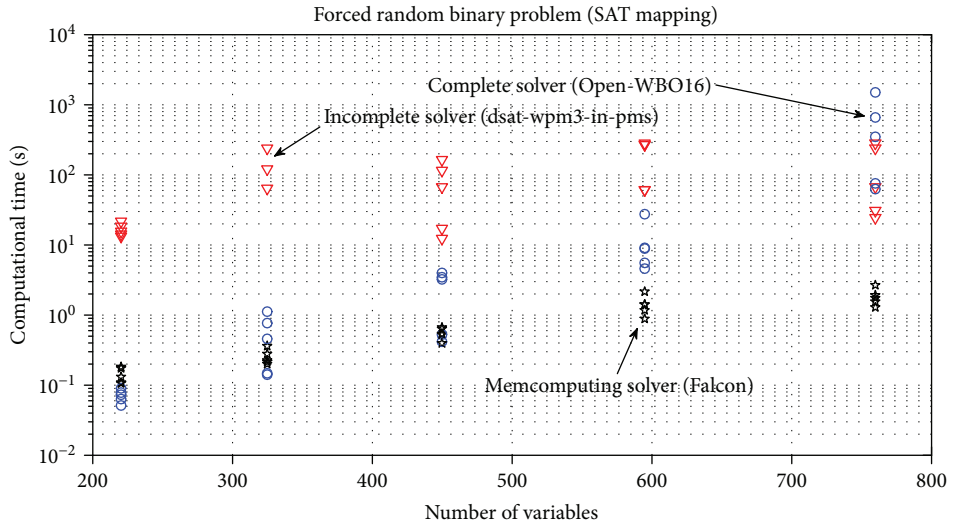


Figure 8: Results from the 2016 Max-SAT competition for the forced random binary problem compared with our memcomputing solver, Falcon.

Because of the presence of hard clauses, the weighted partial Max-SAT is, in general, harder than the Max-SAT for all kind of solvers. In fact, this is one of the main reasons heuristics are often unable to find even approximations to those problems (see, e.g., Figures 8 and 9).

Including weights and hard clauses in self-organizing logic circuits (SOLCs) is simple. Recalling that each OR gate representing a clause has attached at each terminal a dynamic correction module that injects a large current when the gate is in an inconsistent configuration, we can tune the maximum current allowed for each correction module in the following way. We set the maximum current injected by the dynamic correction modules connected to the SO-OR gates proportionally to the weights of the clauses. For the hard clauses, we can set the maximum current injected by the dynamic correction modules connected to the hard SO-OR gates, larger than the sum of all maximum currents injected by the dynamic correction modules connected to all soft SO-OR gates connected to that hard SO-OR gate. This will guarantee that the hard clauses will have always the priority on the soft clauses.

Comparison from the 2016 Max-SAT Competition

We have tested SOLCs on problems taken from the 2016 Max-SAT competition and compared them against the results of the winners of each category of that competition. Even if the comparison is not completely fair because our code is written in MATLAB while the other codes are written in compiled languages, and the benchmark is not the same because we ran on different processors (we ran all our simulations on an Intel Xeon E5-2680 v3 but used the same

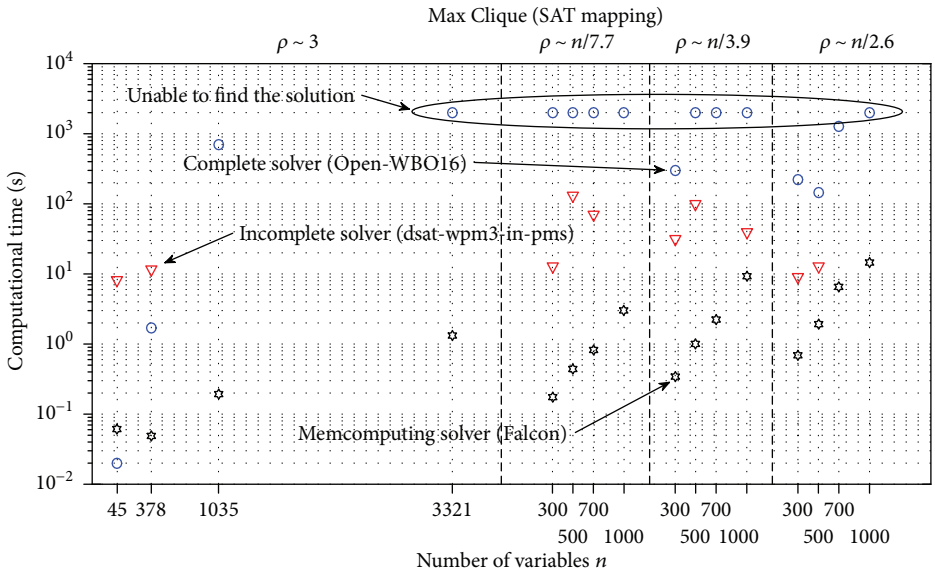


Figure 9: Results from the 2016 Max-SAT competition for the Max Clique problem compared with our memcomputing solver, Falcon.

number of threads allowed in the Max-SAT competition), the results are still interesting.

In Figures 6 and 7, we compare the random Max-2SAT and random Max-CUT instances, which are non-weighted problems [1]. In those cases, the scaling is similar to the heuristics, but the absolute time is orders of magnitude lower.

Of more interest are the results of Figures 8 and 9. These correspond to two problems (called forced random binary and Max Clique [1]) that, when mapped, become weighted partial Max-SAT instances. As discussed, these are especially hard. In fact, oftentimes, the best heuristics could not even find approximations because they were not able to satisfy all hard clauses, while our solver always does.

Data Availability

All calculations reported here have been performed by one of us (Pietro Cicotti) on a single processor of the Comet cluster of the San Diego Supercomputer Center, which is an NSF resource funded under award number 1341698. Apart from the instances freely available from the 2016 Max-SAT competition [6], the authors would be delighted to provide, upon request, all instances of the constrained delta-Max-E3SAT used to generate Figure 2 and those related to all the other figures in this work.

Conflicts of Interest

Fabio L. Traversa and Massimiliano Di Ventra are the cofounders of MemComputing Inc. This company is commercializing the software used in the simulations of this work.

Acknowledgments

The authors sincerely thank Dr. Shaowei Cai for providing the authors with the binary compiled codes CCLS and DeciLS. The authors also thank Haik Manukian and Robert Sinkovits for helpful discussions. Massimiliano Di Ventra and Fabio L. Traversa acknowledge partial support from the Center for Memory Recording Research at UCSD. Massimiliano Di Ventra and Forrest Sheldon acknowledge partial support from the MemComputing Inc.

References

[1] K. S. Christos and H. Papadimitriou, *Combinatorial Optimization*, Dover Publications Inc., 1998.

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

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

[4] H. Manukian, F. L. Traversa, and M. Di Ventra, *arXiv*, vol. 1801, article 00512, 2018.

[5] M. Di Ventra and F. L. Traversa, *Journal of Applied Physics*, vol. 123, no. 18, article 180901, 2018.

[6] http://www.maxsat.udl.cat/16/index.html.

[7] C. P. Gomes, H. Kautz, A. Sabharwal, and B. Selman, *Handbook of Knowledge Representation*, F. Van Harmelen, V. Lifschitz, and B. Porter, Eds., vol. 1, Elsevier, 2008.

[8] H. A. Kautz, A. Sabharwal, and B. Selman, *Handbook of Satisfiability*, A. Biere, M. Heule, and H. van Maaren, Eds., vol. 185, IOS press, 2009.

[9] J. Hromkovic, *Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics*, Springer, 2010.

[10] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[11] J. Hastad, "Some optimal inapproximability results," *Journal of the ACM*, vol. 48, no. 4, pp. 798–859, 2001.

[12] F. L. Traversa and M. Di Ventra, "Polynomial-time solution of prime factorization and NP-complete problems with digital

memcomputing machines," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 2, article 023107, 2017.

[13] F. L. Traversa and M. Di Ventra, "Universal memcomputing machines," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 11, pp. 2702–2715, 2015.

[14] M. Di Ventra and Y. V. Pershin, "The parallel approach," *Nature Physics*, vol. 9, no. 4, pp. 200–202, 2013.

[15] M. Di Ventra, F. L. Traversa, and I. V. Ovchinnikov, "Topological field theory and computing with instantons," *Annalen der Physik*, vol. 529, no. 12, article 1700123, 2017.

[16] F. L. Traversa, Y. V. Pershin, and M. Di Ventra, "Memory models of adaptive behavior," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 9, pp. 1437–1448, 2013.

[17] F. L. Traversa, F. Bonani, Y. V. Pershin, and M. Di Ventra, "Dynamic computing random access memory," *Nanotechnology*, vol. 25, no. 28, article 285201, 2014.

[18] C. Luo, S. Cai, W. Wu, Z. Jie, and K. Su, "CCLS: an efficient local search algorithm for weighted maximum satisfiability," *IEEE Transactions on Computers*, vol. 64, no. 7, pp. 1830–1843, 2015.

[19] S. Cai, C. Luo, and H. Zhang, "From decimation to local search and back: a new approach to MaxSAT," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, pp. 571–577, Melbourne, Australia, August 2017.

[20] B. Selman, H. Levesque, and D. Mitchell, "A new method for solving hard satisfiability problems," in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 440–446, San Jose, CA, USA, July 1992.

[21] S. Kirkpatrick and B. Selman, "Critical behavior in the satisfiability of random boolean expressions," *Science*, vol. 264, no. 5163, pp. 1297–1301, 1994.

[22] S. Cocco, R. Monasson, A. Montanari, and G. Semerjian, *Analyzing Search Algorithms with Physical Methods*, Computational Complexity and Statistical Physics, 2006.

[23] W. Barthel, A. K. Hartmann, M. Leone, F. Ricci-Tersenghi, M. Weigt, and R. Zecchina, *Physical Review Letters*, vol. 88, no. 18, article 188701, 2002.

[24] F. Ricci-Tersenghi, M. Weigt, and R. Zecchina, "Simplest random K-satisfiability problem," *Physical Review E*, vol. 63, no. 2, 2001.

[25] H. Jia, C. Moore, and B. Selman, *International Conference on Theory and Applications of Satisfiability Testing*, Springer, 2004.

[26] M. Mezard and A. Montanari, *Information, Physics, and Computation*, Oxford University Press, 2009.

[27] F. L. Traversa, C. Ramella, F. Bonani, and M. Di Ventra, "Memcomputing NP-complete problems in polynomial time using polynomial resources and collective states," *Science Advances*, vol. 1, no. 6, article e1500031, 2015.

[28] J. Hale, "Asymptotic behavior of dissipative systems," in *Mathematical Surveys and Monographs*, vol. 25, American Mathematical Society, Providence, Rhode Island, 2nd edition, 2010.

[29] M. Di Ventra and F. L. Traversa, *Physics Letters A*, vol. 381, no. 38, pp. 3255–3257, 2017.

[30] M. Di Ventra and F. L. Traversa, "Absence of periodic orbits in digital memcomputing machines with solutions," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 10, article 101101, 2017.

[31] J. S. R. Bulirsch, *Introduction to Numerical Analysis*, Springer, 2010.