scientific reports

Check for updates

OPEN

Combinatorial optimization by weight annealing in memristive hopfield networks

Z. Fahimi[1,3]✉, M. R. Mahmoodi[1,3]✉, H. Nili[1], Valentin Polishchuk[2] & D. B. Strukov[1]

The increasing utility of specialized circuits and growing applications of optimization call for the development of efficient hardware accelerator for solving optimization problems. Hopfield neural network is a promising approach for solving combinatorial optimization problems due to the recent demonstrations of efficient mixed-signal implementation based on emerging non-volatile memory devices. Such mixed-signal accelerators also enable very efficient implementation of various annealing techniques, which are essential for finding optimal solutions. Here we propose a "weight annealing" approach, whose main idea is to ease convergence to the global minima by keeping the network close to its ground state. This is achieved by initially setting all synaptic weights to zero, thus ensuring a quick transition of the Hopfield network to its trivial global minima state and then gradually introducing weights during the annealing process. The extensive numerical simulations show that our approach leads to a better, on average, solutions for several representative combinatorial problems compared to prior Hopfield neural network solvers with chaotic or stochastic annealing. As a proof of concept, a 13-node graph partitioning problem and a 7-node maximum-weight independent set problem are solved experimentally using mixed-signal circuits based on, correspondingly, a 20 × 20 analog-grade TiO_2 memristive crossbar and a 12 × 10 eFlash memory array.

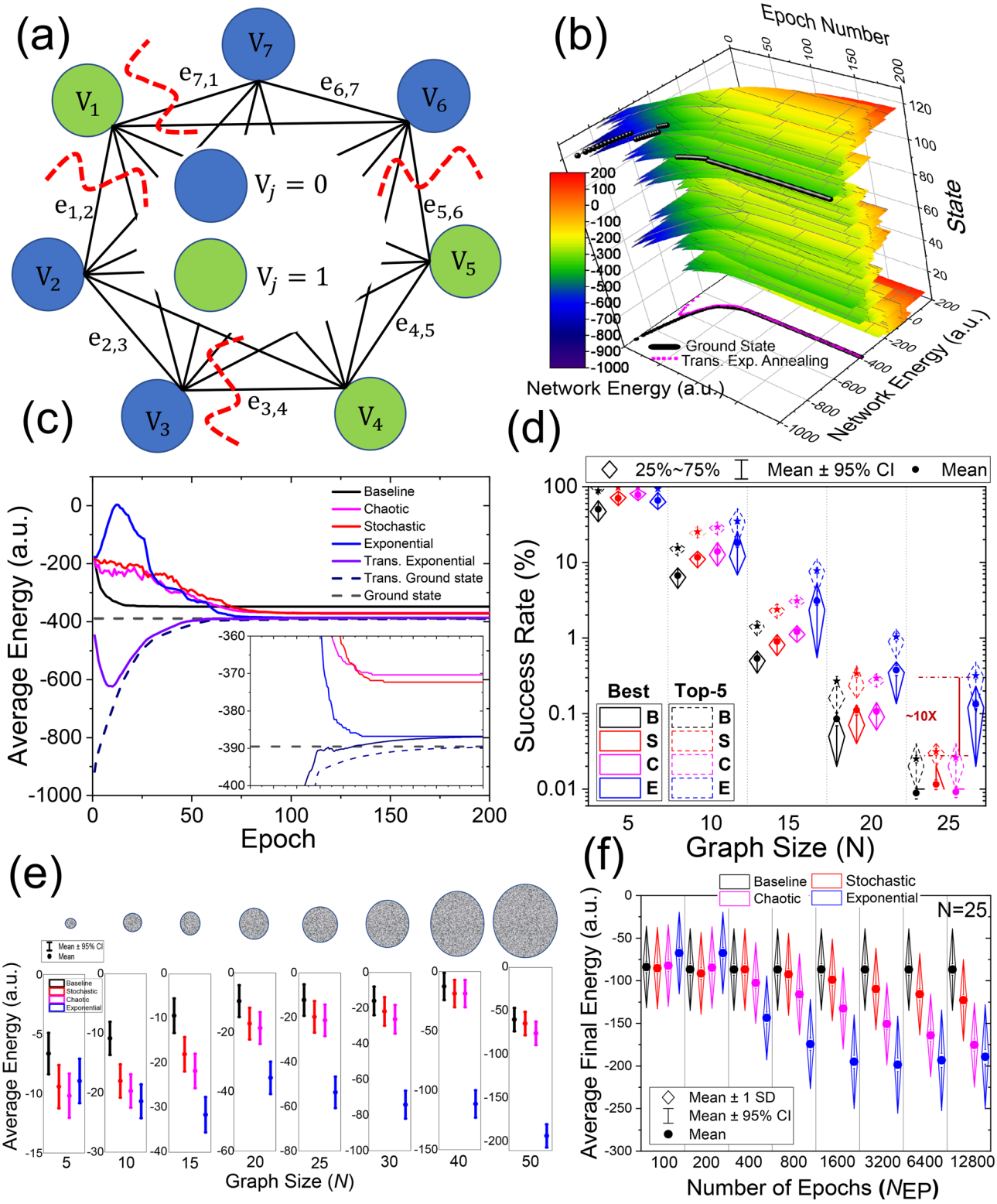
Combinational optimization is an essential subset of mathematical optimization methods with numerous applications in various fields, including operation research, machine learning, and scientific computing[1–3]. A typical goal of combinatorial optimization is to find an optimal solution within a finite set of possible solutions. For example, graph partitioning, that is, the problem of minimizing the cutsize when partitioning a graph into two sections of nearly equal weight, finds applications in distributed computing and digital VLSI design flow.

For most combinatorial problems, the exhaustive brute-force search is often not practical, and developing efficient heuristic and meta-heuristic methods is of utmost importance[4,5]. The enormous computational power required to solve large-scale optimization problems also poses a great challenge. The problem is exacerbated by the sequential structure of general-purpose processors, which are very energy-demanding and inefficient in running large-scale, massively parallel algorithms. Hence, hardware accelerators, e.g., based on superconductors[6,7], digital CMOS[5,8,9], nanomagnetic[10], and photonic [11] technologies, are proposed to solve optimization problems using heuristic methods efficiently.Hopfield neural network (HNN)[12–14] is also a heuristic method that extended the application of neural networks from classification to optimization and associative memory. A particular class of recurrent HNNs is the discrete-time asynchronous model, which operates based on a single neuron update at a time mechanism. For a network featuring N binary neurons, a randomly-selected jth neuron is updated at time $t + 1$ using

$$U_j(t+1) = f\left(\sum_{i=1}^{N} w_{ij}(t)U_i(t) + T_j^b\right), \tag{1}$$

where $U_j(t)$ is the binary state of the jth neuron at time t, $w_{ij}(t)$ is the synaptic strength between neurons i and j at iteration t, T_j^b is the bias strength of the jth neuron, and f(.) is the binary threshold function. The key features of HNNs are their activation dynamics and energy function, which are proven to be monotonically descending during the runtime[15] (see Supplementary Section 1 for more details). Hence, by mapping the cost function of the optimization problem into the energy of the network and the variables to neuron states, the recurrent dynamic of the network optimizes the cost function and solves the optimization problem in the runtime.

[1]UC Santa Barbara, Santa Barbara, CA 93106-9560, USA. [2]Linkoping University, 60174 Norrköping, Sweden. [3]These authors contributed equally: M. R. Mahmoodi and Z. Fahimi. ✉email: z.fahimi@ucsb.edu; mrmahmoodi@ucsb.edu


(a)
V7
V1
V6
V2
V5
V3
V4
e7,1
e6,7
e1,2
e5,6
e2,3
e4,5
e3,4
Vj = 0
Vj = 1
(b)
Epoch Number
State
Network Energy (a.u.)
Ground State
Trans. Exp. Annealing
(c)
Average Energy (a.u.)
Epoch
Baseline
Chaotic
Stochastic
Exponential
Trans. Exponential
Trans. Ground state
Ground state
(d)
25%~75%
Mean ± 95% CI
Mean
Success Rate (%)
Graph Size (N)
Best
Top-5
~10X
(e)
Average Energy (a.u.)
Graph Size (N)
(f)
Average Final Energy (a.u.)
Number of Epochs (NEP)
N=25
Mean ± 1 SD
Mean ± 95% CI
Mean

◂**Figure 1.** Neuro-optimization with the weight annealing: (**a**) The 7-node weighted graph partitioning problem that is used to illustrate the mechanism of weight annealing. The blue/green coloring shows the optimum solution (Supplementary Materials S3 includes the actual weights). (**b**) The energy evolution of each state during weight annealing for 200 epochs and $\tau = 40$. The black spheres mark the transitory ground state of the system, which is also projected to the energy-epoch plane. The magenta curve shows the average transitory energy over 128 runs, which shows that the proposed weight annealing tracks the transitory globally optimum state of the system. (**c**) The average energy of the network annealed with different techniques over 128 runs. (**d**) Top-1 and Top-5 success rates of varying annealing techniques versus problem size (B: baseline, i.e., the standard Hopfield network without annealing, S: stochastic (temperature reduced from 100 to 0.01), C: chaotic (temperature reduced from 250 to 0.001), and E: exponential weight annealing). For each graph size, we consider 200 randomly weighted problems and provide the parameters in supplementary S4. Note that the best response is the global optimum, and Top-5 counts if the final response is among the best top-5 solutions. Panel (**e**) shows the distribution of the final average energy, offset by a constant for clarity, for the same graphs used in panel (**d**). The circles represent graph size. (**f**) The boxplot of the average final energy vs. epoch size for 200 random configurations of 25-node graph partitioning problems.

Similar to the Ising model and other greedy and local search methods, the critical shortcoming of HNN is the presence of (many) local minima in their energy function. Simulated[16,17] and chaotic[18–20] annealing are two prominent techniques that tackle this issue by harnessing thermally controlled probabilistic jumps and embedded chaos in HNNs with nonzero self-feedback weights, respectively. Therefore, an efficient HNN accelerator should perform the frequent dot-product operation in Eq. 1 very fast and support an annealing technique to rescue the network from trapping in local minima. This paper introduces a weight annealing technique in HNNs and its efficient implementation, which is more effective and scalable than simulated and chaotic annealing methods. Our approach dates back to methods like weight annealing[21–24], noising[25], space smoothing[26,27], and fine-tuned learning[28], where the core idea is to change in the energy landscape by modifying weights in the formula for the energy. Here, the exact meaning of "weight" varies from method to method, as well as from problem to problem addressed—a weight may be associated with an input data point, a subproblem, etc.; similarly, a variety of ways to modify the weights (random perturbation, adversarial change, etc.) has been explored. The common crux of the methods is that they modify the weights differently in every timestep and in different areas of the solution space; this way, the search is guided by weight changes adapted to the current state and reuses insights gained from previous iterations. While the clever schemes for such adaptive weight modifications underpin the strengths of methods, mimicking this adaptivity within any hardware would likely be inefficient since performing individual changes to the weights consumes significant time and energy. Further, hardware implementation of the algorithms that act differently in different parts of the solution space would require complicated circuitry, leading to efficiency losses. Our proposed weight annealing circumvents both of the above: First, all weights are scaled together at every iteration. Second, the weight modification is oblivious to the status of the solution space exploration—the annealing schedule is pre-set in advance and does not depend on the state of the system (in particular, the schedule does not depend on the value of the energy function—it is the hardware that takes care of the derivatives, convergence, escaping local optima with stochastic decisions, etc.). We numerically demonstrate the effectiveness of our approach on several benchmarks by solving graph partitioning, vertex cover, maximum-weight independent set, and maximum-weight clique problems.

We also propose a very efficient implementation of weight annealing in HNNs harnessing analog-grade non-volatile memories, which have become the mainstream devices for implementing fast, compact, and energy-efficient dot-product engines[29–32]. The potentials for performing high-speed physical-level computing are perhaps the most intriguing feature of these devices. Passive (0T1R) memristive devices are the most promising candidate for the next generation of analog computing systems in part due to their excellent scalability prospects and superior integration density[33–36]. Furthermore, recent breakthroughs in exploiting embedded eFlash memories have opened the doors towards building large-scale industrial-grade neurocomputing systems[32,37] as well. These exciting opportunities have served as the motivations for several experimental proposals on Hopfield networks, simulated annealing, and related concepts.

Reference[38] uses discrete $Pt/TiO_{2-x}/Pt$ memristive devices to implement a small-scale 4-bit data converter with the Hopfield model. Reference[39] implements a 3-bit associative memory based on digital HfO_2 memristors. In Ref.[40], simulation results demonstrate the effectiveness of using the inherent chaos in sub-100 nm NbO_2 memristors to implement simulated annealing within Hopfield networks. Ref. [41] implements an 18-node restricted Boltzmann machine based on a versatile stochastic dot-product engine using TiO_2 memristive crossbars[42]. In addition, Ref. [41] demonstrates hardware implementation of simulated, chaotic, and adjustable annealing within HNNs. Conceptually, the proposed weight annealing is similar to the adjustable technique as it relies on dynamic scaling of the energy during runtime. However, the proposed method has a more straightforward implementation as it does not require extra circuitry, is not limited to the dynamic range of devices, and can be generally applied to any HNN irrespective of the target optimization problem. Several works (e.g., see[43–45]) propose using the inherently random switching mechanism of memories to implement stochastic sigmoid neuron functionality and simulated annealing. However, this method suffers from the limited switching endurance, cycle-to-cycle and device-to-device variations, and scalability issues. Finally, Ref. [46] uses Y-flash memories to implement a 3-bit associative memory based on the Hopfield model.

Results

The proposed idea is to slowly modulate the energy landscape of the HNN, starting from a funnel shape with a deep global optimum where the ground state is easily accessible. The network traps in it in the early stages and tends to remain in ground states during the runtime. In our proposed method, we change the synaptic weights slowly by considering $w_{ij} = T_{ij}\left(1 - e^{\frac{-t}{\tau}}\right)$ where $\tau > 0$ is the annealing schedule, and T is the ultimate synaptic weight matrix. The Lyapunov energy associated with a certain state of the network at t is given by

$$E(t) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(t)U_i(t)U_j(t) - \sum_{i=1}^{N} T_j^{\mathrm{b}}U_j(t). \tag{2}$$

At the beginning and while $t<<\tau$ is very small, the first term [in Eq. (2)] is negligible, and the total energy of the network is $-\sum_{i=1}^{N} T_j^b U_j(t)$. At this stage, the network finds a straightforward solution after few updates. The ground state, for example, is located at $U_j = 1$ for the jth neuron that has $T_j^{\mathrm{b}} > 0$. As the network evolves, $w_{ij}(t)$ gradually moves toward $T_{ij}(t)$ and the first term in Eq. (2) becomes more significant until the network stabilizes in the equilibrium state. During this runtime, the ground state of the network changes many times, but the network tends to capture it and closely follows the transitory ground state.

We consider graph partitioning problems (see Supplementary Section 2) that find applications, e.g., in graph-based electronic structure theory applied to quantum molecular dynamic simulations[45]. To demonstrate a clear visual representation of ground state evolution, we use a 7-node graph partitioning problem with randomly selected weights and edges, as shown in Fig. 1a (see Supplementary Sect. 3 for the actual vertex and edge weights). Figure 1b shows the semi-exponential energy change of all possible states during the annealing ($\tau = 40$). The energy associated with each state is exponentially increasing as expected. The black sphere points (projected to the bottom plane for clarity) represent the ground state of the system during the annealing. The global optimum is −389.5459 and locates at state 97 (decimal equivalent of "1,100,001"). The transitory state of the system is specified by listing the N values of U_j and represented by a binary word of N bits[14] and its decimal version for simplicity. At $t = 0$, the global minimum is recognizable (state 118, $E = -917.76$). While the network is steadily evolving, the ground state of the system increases, and its location changes several times. The average transitory energy of the system (defined over the transitory synaptic weights) is also shown for 128 initialization schemes and 200 epochs ($N_{\mathrm{EP}} = 200$) in magenta. The network finds the initial ground state very quickly (regardless of the initial state) owing to the annealing mechanism and tracks it during the evolution. Other simulation details, including w_{ij} evolution are provided in Supplementary Sect. 3.

Figure 1c shows the performance of the proposed annealing technique versus stochastic annealing with a probabilistic sigmoid neuron (the temperature is reduced exponentially from 100 to 0.01) and chaotic annealing (the self-feedback weights are decreased exponentially from 250 to 0.001). In this experiment and after 200 epochs, the success rate (the relative number of cases led to the global optima) is 57.8%, 59.37%, 94.53% for chaotic, stochastic, and weight annealing techniques, respectively, and it is 28.12% for the standard Hopfield model (baseline). It is noteworthy that the stochastic annealed network converges to $E = -387.98$ and scores a 98.6% success rate when 30 k epochs are used, and the temperature is scaled from 100 k to 0.01.

To further investigate the performance of the proposed approach, 200 randomly populated configurations of 5, 10, 15, 20, and 25-node graphs are considered. Supplementary Sect. 4 discusses the parameters used in the simulations. The annealing schedule parameter is manually optimized for the first problem and used in all configurations. The scalability of our approach is compared with simulated annealing on three scenarios: first, $N_{\mathrm{EP}} = 300$ is assumed for all sizes, then it is exponentially increased for a fixed-size graph ($N = 25$), and then, N_{EP} is exponentially increased with respect to the linear increase of the problem size.

The success rate achieved by different methods on various problem sizes for $N_{\mathrm{EP}} = 300$ is shown in Fig. 1d. The performance of weight annealing is on par with simulated annealing for $N = 5$; however, the energy gap becomes significantly wider for larger problem sizes. More interestingly, for $N = 15$, among the 200 configurations, the 20 percentiles success rate of weight annealing is better than the 80 percentiles of all other methods. Note that due to the analog-grade behavior of our memristors, weighted graph problems are considered, and it would be unfair to compare our results (in terms of success rate) with previous implementations, which focus on sparse graphs with binary weights. Figure 1e shows the average final energy for the same graphs. The gap between the solution quality (final energy) of exponential weight annealing and other methods becomes wider in more massive graphs. In Fig. 1f, the computational runtime (epoch number) is increased for 200 configurations of 25-node graphs. As expected, the performance of all annealing techniques, including weight annealing, improves by increasing the number of epochs (in part due to slower cooling, which allows the networks to search for better solutions). The performance of weight annealing no longer improves for $N_{\mathrm{EP}} > 3200$, while simulated annealing techniques, with noticeable inferior performance, benefit from the longer computational time and slower annealing. This is partly due to the inherent differences between the underlying mechanism of simulated and exponential weight annealing. Stochastic annealing requires more time to explore larger searching spaces. While for the weight annealing, it is simply not the case. The accuracy saturation stems from the fact that the slower learning of weights no longer creates a more optimum path. Note that weight annealing achieves the same solution quality 10 × faster than simulated annealing techniques. Supplementary Sect. 4 extends the graph partitioning simulations. Three other combinatorial optimization problems are considered in Supplementary Sect. 5, and the results signify the superiority of weight annealing, particularly in large scale problems.

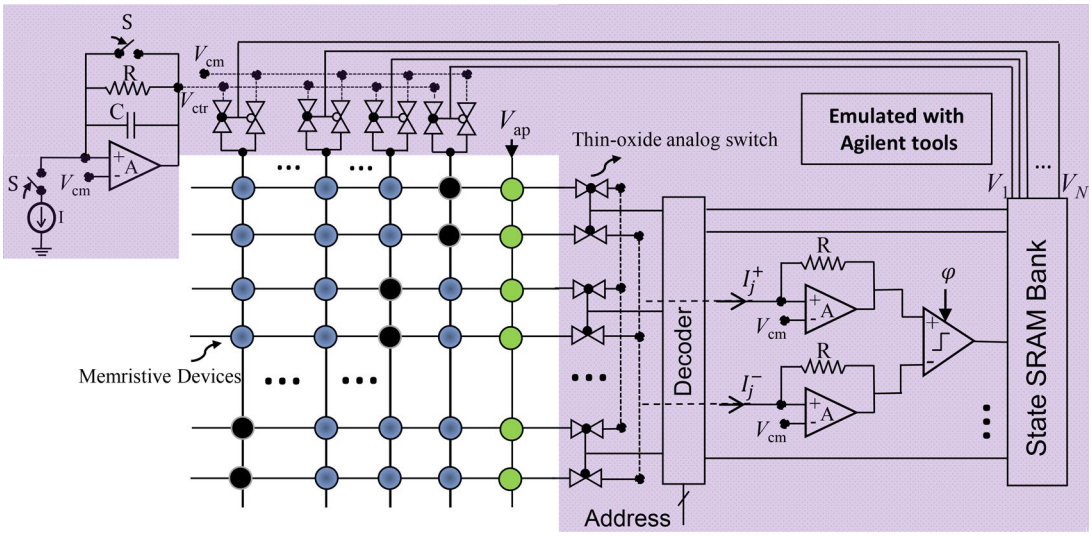


Figure 2. The current-mode recurrent circuit that implements the weight annealing of discrete-time Hopfield networks with programmable analog memories. The green circles show the bias weights (T_i^b) while the black circles implement self-feedback weights (T_{ii}), and the rest of them denote the main synaptic weights ($T_{ij,i\neq j}$). A constant 'on' voltage, which is the same as the tuning voltage, drives the bias column. We control the applied voltage to the rest of the devices during the runtime to adjust the synaptic weights (exponentially). Note that V_{cm} is only added to emphasize that the circuit operate on a single-V_{dd}. Values R, C, and I depend on the problem size and technology, and determine the annealing schedule. Switch S resets the network to the initial condition. The selected neuron is determined by the input address to the decoder, and the operation is synchronized with the sampling clock (φ) in dynamic comparator. Note that we have omitted the tuning circuits in the figure for clarity.

Experimental results

The proposed technique is demonstrated by addressing two optimization problems based on the most prospective analog-grade memory technologies. The central merit of weight annealing lies in its very straightforward and compact implementation. Experimental results of hardware implementation are demonstrated by solving a 16-node graph partitioning problem using a 20×20 passively integrated analog-grade memristive crossbar and a 7-node maximum-weighted independent set on a 12×10 embedded array of eFlash memories.

Figure 2 shows the implementation of the weight annealing technique. The corresponding hardware realization of Eq. (1) is discussed in the method section for both cases. The main challenge in realizing the weight annealing is scaling the synaptic weights. Let us emphasize that direct modification of (analog) states is impractical in part because of the limited endurance, device-to-device, and cycle-to-cycle variations. This challenge can be resolved in resistive memories by using a simple control circuit (the pre-synaptic drivers), which scales all synaptic weights simultaneously (see Fig. 2). Here, V_{ctrl} is exponentially increased toward V_{ap} at which all devices are tuned. The current neuron state determines which devices should be driven by V_{ctrl}. The post-synaptic circuits include trivial circuits such as transimpedance amplifiers (e.g., a buffered version of Ref.[47]) that senses currents and a dynamic voltage comparator (see, e.g.,[48]) that updates the selected neuron state. These circuit functionalities are emulated with Agilent characterization tools in the present demonstration.

In split-gate embedded Flash memories, the situation is more straightforward as we can bias the memories in the weak inversion regime, making their states (i.e., currents) semi-exponentially dependent on the select-gate voltage. Then, V_{ctrl} is applied to the shared select-gates and linearly increased toward the V_{ap}.

In the first experiment, a 13-node graph partitioning problem is implemented using passively-integrated memristive crossbars. Note that, to the best of our knowledge, this work is the largest Hopfield network implemented with passive memristors. Figure 3a shows the wire-bonded chip, crossbar TEM image, and an SEM image of a memristive device. This crossbar has been previously used for the demonstration of a multilayer perceptron[49], integrated spiking neural network for coincident detection[29], and a hardware security primitive design[50,51]. The method section includes a brief description of fabrication steps. More relevant details are also available in our previous work[49].

In order to increase the demo size and without the loss of generality, we have ensured the weights and edges (of the graph) are selected such that $T_{ij}<0$ and $b_j>0$ (see Supplementary Sect. 6 for more details). This facilitates a single-ended time-multiplexed dot-product of a $13\times(13+1)$ network on our memristive crossbars. The details of forming, tuning, and operation of the circuit, as well as the procedure of mapping the actual synaptic weights (from software) to conductance values, are illustrated in the method section. After determining the desired conductance map, the devices are programmed individually using the write-verify algorithm[54]. Figure 3b,c show the desired weight map of the network and the corresponding conductance map obtained after tuning the crossbar, respectively. Most devices are tuned very close (within 5%) to the desired states, which is possible due to the tight distribution of switching thresholds in our analog-grade crossbar circuits. Figure 3d shows the distribution of pre-activation readout currents for the baseline case (the inset indicates no bias in neuron

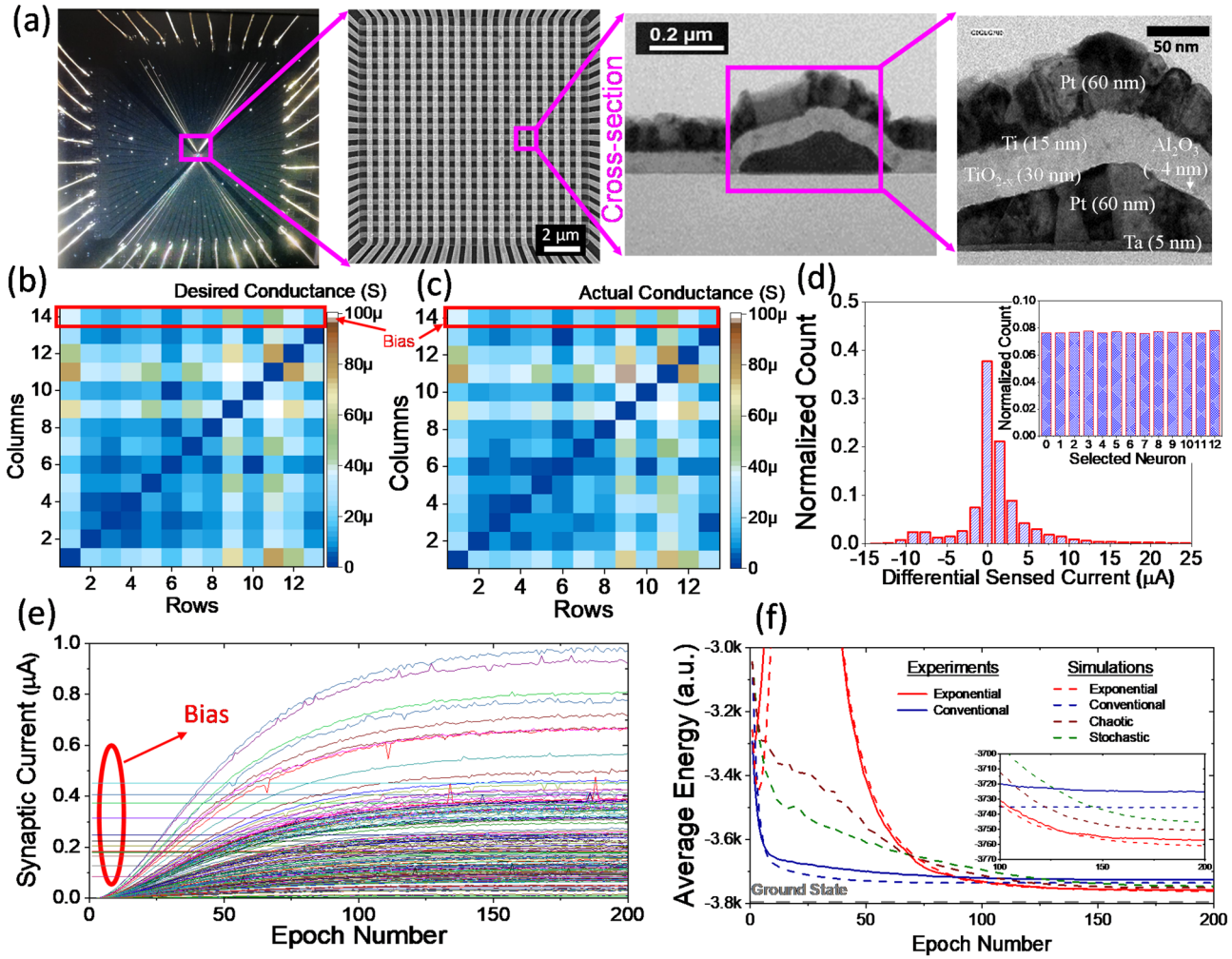


Figure 3. The experimental demonstration with the integrated memristor crossbars. (**a**) The fabricated 20 × 20 integrated memristor crossbar[29]. (**b**) The desired ideal analog map for the 13-node graph partitioning problem, and (**c**) the resultant conductance map of the devices after tuning the crossbar. (**d**) Distribution of the readout current when solving the problem with the conventional (baseline) approach. The inset shows the histogram of selected neurons (for updates) and indicates there is no bias in the neuron update. (**e**) The evolution of the synaptic weights during the weight annealing. (**f**) The experimental versus simulation results of the neuro-optimization with different techniques. The inset shows the zoomed-in average energy in the last 100 epochs.

selection). The input "on" voltage corresponding to binary input '1' is $V_{ap} = 0.1$ V. Note that we exponentially increase the "on" applied voltage from 0 to 0.1 V for the weight annealing. The measured synaptic strength of each device during the weight annealing is shown in Fig. 3e. The experimental and simulation results are compared in Fig. 3f. Specifically, the average energy over 10^3 cases for 200 epochs is shown for various methods. Here, the annealing schedule parameters are 10^4, 10^5, and 35 for chaotic, stochastic, and weight annealing, respectively. The ground state locates at − 3796, and weight annealing (on both experiment and simulation) performs better than other techniques and far better than the baseline.

In our second experimental demo, a 7-node maximum-weighted independent set is solved using an array of 12 × 10 redesigned embedded Flash memories fabricated in Global Foundries 55 nm LPe CMOS process (Fig. 4a). The redesigned array structure enables < 1% analog programmability[52] (see Fig. 6S). The circuit diagram in Fig. 6Sd implements the weight annealing of Hopfield networks with eFlash memories. Biasing conditions (imposed during programming) ensure the subthreshold operation of the devices at all operating conditions. Figure 4b shows the implemented weighted graph. Similar to the first demo, the weights and edges (of the graph) are chosen randomly but constrained by $T_{ij} < 0$ and $T_j^b > 0$. The original weight matrix is shown in Supplementary Sect. 7. The ground state of the energy function locates at -5.5755 that corresponds to the neural state "0010001".

The devices are programmed with < 1% accuracy (see the method section). Figure 4c shows the resultant map of state currents under nominal biasing conditions, i.e., ($V_{WL} = 1.5$V, $V_{CG} = 2.5$V, $V_{BL} = 1$V, $V_{SL} = 0$V, and $V_{EG} = 0$V). The experiments and simulations are performed over 128 initialization cases for 500 epochs and show the results in Fig. 4d. The results are averaged over 100 runs in the simulations. The annealing schedule is 10, 10, and 100, and the average probability of hitting the global optimum is 0.76, 0.92, 0.82, and 0.99 for stochastic, chaotic, and weight annealing, respectively (Fig. 4e). We drive the devices

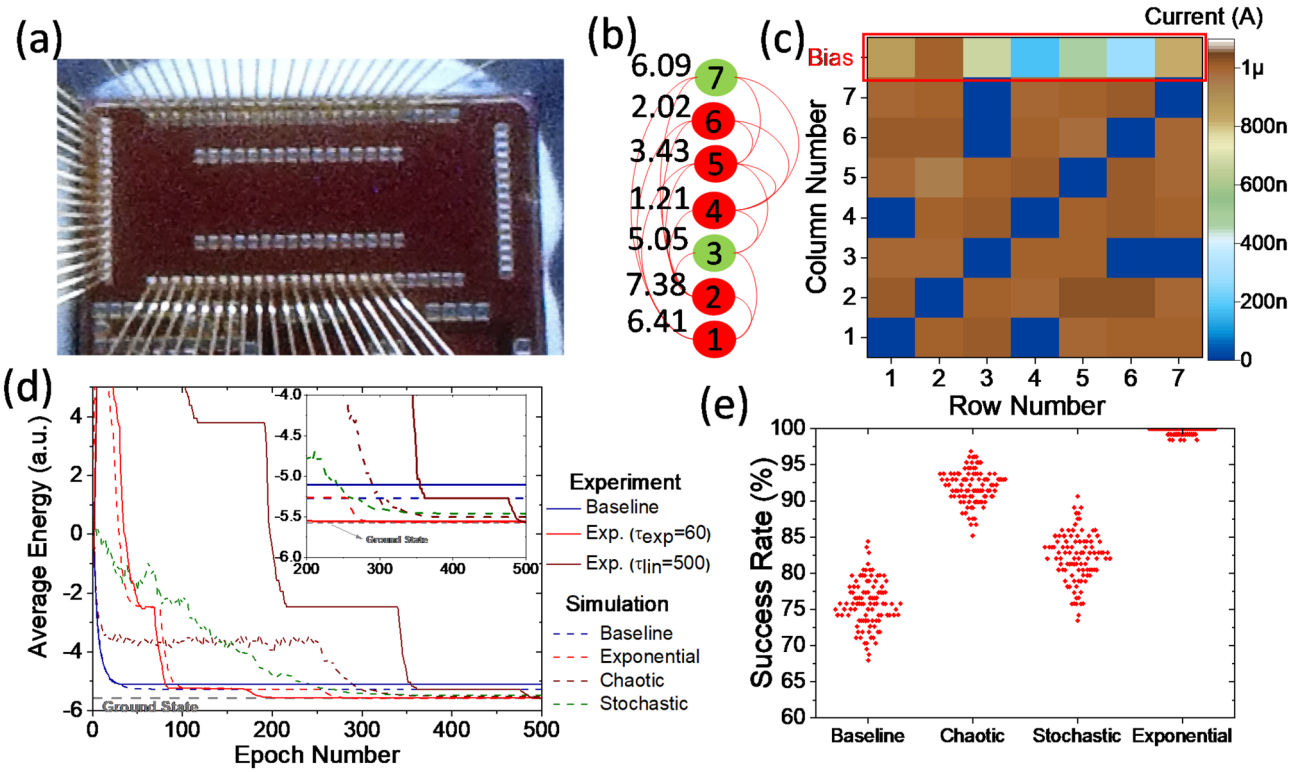


Figure 4. Neuro-optimization with embedded analog-grade eFlash memories. Panel (**a**) shows the fabricated 10 × 12 eFlash array chip in Global Foundries' standard LPe CMOS process[52]. (**b**) A 7-node maximum-weighted independent set problem. (**c**) The heat map of the synaptic weights for the devices that implement the neuron-optimization. (**d**) The average energy versus epoch comparing experimental results with simulations over 100 runs. (**e**) The success rate of different annealing techniques on this problem over 100 runs.

corresponding to bias weights (T_j^b) by constant gate-voltages ($V_{WL} = 1.5V$ and $V_{CG} = 2.5V$), while other rows (if their corresponding neuron is in the 'on' state) are driven by $V_i(t)$. The impact of annealing schedule and exponential versus linear voltage scaling are also studied in Fig. 7S. For the former, a slower annealing schedule ($\tau_{exp} = 60$) tackles the nonlinearities in the super-exponential dependency of synaptic current to voltage and closely match the trends in the simulations. For the latter case, the slowest annealing process ($\tau_{exp} = N_{EP} = 500$) leads to the best response.

Discussion and summary

We have demonstrated weight annealing, a technique that improves the performance of asynchronous Hopfield neuro-optimizer. The weight annealing converges faster and to a better solution within studied runtime as compared to other considered annealing approaches. The scalability of weight annealing (size and computational time) is investigated by solving several combinatorial problems, and its straightforward implementation is demonstrated the using two state-of-the-art analog-grade non-volatile memories.

The passive integrated memristor technology offers the best scaling prospects and low fabrication cost. We have recently developed a 4 K fully CMOS-compatible 0T1R array with excellent switching characteristics[36]. The measured analog characteristics are promising for the development of large-scale neuro-optimization systems. On the other hand, eFlash technology is much sparser, but it is currently commercially available and embedded in standard CMOS processes (down to 28 nm). Our preliminary estimations (see Supplementary Sect. 8 for more details) indicate impressive prospects of using metal-oxide memristors for the hardware implementation of Hopfield networks and weight annealing. Future works focus on the CMOS-integrated design of a weight annealing optimizer, allowing us to perform a rigorous comparison with entirely fabricated annealing machines.

As opposed to most previous works[43–45] that focus on switching statistics of memristors, our proposed solution offers very infrequent writes, which is justified assuming long runtimes of computationally extensive problems. More importantly, our proposed neuro-optimizer offers analog (> 5 bits with memristive nanodevices and > 6 bits via eFlash technology) weights. This feature is not demonstrated in most previous Ising machines. Unlike quantum computing machines that are susceptible to environmental noise, hard to scale, and must operate at cryogenic temperatures, the proposed circuit is more scalable, and can operate at room temperatures.

In summary, the proposed weight annealing boosts the performance of HNN in solving combinatorial optimization problems. Using extensive simulations on four representative problems, we numerically demonstrate that the proposed method outperforms the conventional Hopfield network (baseline) and challenges the prominent stochastic and chaotic annealing techniques in computational time and accuracy. Then, an efficient, scalable,

and fast circuit implementation and experimentally verified based on two memory technologies. Large-scale integrated implementation is demonstrated of weight annealing is a near-term future work.

Methods

In the first experiment, we demonstrate the weight annealing with a 20×20 array of passively integrated crossbars of 600-nm pitch memristive devices (200-nm lines separated by 400-nm gaps) fabricated in the University of California at Santa Barbara's nanofabrication facility. The fabrication and characterization details are discussed in [29,49]. In summary, we deposit the active bilayer by low-temperature reactive sputtering, evaporate electrodes using oblique angle physical vapor deposition, pattern them by lift-off technique, and then contact them to bonding pads. The crossbar is wire-bonded in a dual in-line package and mounted on a custom-made printed circuit board, as shown in Supplementary Section 9.

The devices are in pristine states upon fabrication and require electroforming to become programmable devices. An automated setup performs the current-controlled electroforming process device per device. A compliance voltage (1.5 V to begin with, but it is dynamically updated) prevents the memristors from burning. For every device, we sweep the applied current from 0 to 100 μA and monitor its resistance consistently. The process continues until the device reaches an acceptable low resistance (typically 5 kΩ–150 kΩ). The devices are formed individually and reset them after each forming success (to remove leakage for the rest of the crossbar). A dynamic leakage removal procedure is also employed to reset the devices when the algorithm struggles to form several devices in a row.

The devices are tuned using an ex-situ approach meaning that weights (T_{ij}) and biases (T_b) are obtained from software simulations and later transferred to the crossbar. Indeed, after forming the entire crossbar, i.e., the 400 devices (yield is typically > 99%), the memristors are tuned to the desired states individually using V/2 and write-verify schemes. The automated algorithm progressively increases the pulse amplitude from 0.5 to 2 V (to increase the conductance) and from 0.5 to 2.2 V (to reduce it). The pulse width is 1.1 ms during the programming. Each device typically needs ~ 50 pulses to reach within 2% of the targeted state. The fabricated crossbar has a reasonably uniform and tightly distributed switching thresholds ranging from 0.6 to 1.5 V (for set) and − 0.6 to − 1.7 V (for reset), which provides us with the opportunity to harness the V/2 scheme and precisely tune the devices. The devices have excellent retention characteristics, and accelerated retention tests report minor < 1% change in after the projected 10 years of operation at room temperature. Additional details are provided in Ref.[22].

In order to increase our demo size (given our 20×20 crossbar size), we deliberately chose edges to be larger than weights (the values are selected randomly in all experiments and simulations) to force all non-diagonal synaptic weights (T_{ij}) to be negative and all biases to be positive. This technique allows us to implement a relatively larger demo by assigning one device per weight (in comparison with the two-device per synapse needed for fully differential design) and perform each the vector-by-vector multiplication in two cycles. Indeed, the dot-product operation is implemented in a two-step time-multiplexed fashion; that is, in one cycle, we measure the total current ($\sum I^-$) associated with the input vector multiplied by the synaptic weight vector (from the selected neuron), while the input bias voltage is zero. Then, we subtract it from the sensed current ($\sum I^+$) from the same bitline, while the main inputs are zero and apply $V_{ap} = 0.1$ V to the bias column. Besides, to increase the dynamic range, all bias conductances are divided by 5 and compensated by applying an extra gain of 5 at the neuron side. In other words, the final output is evaluated by hard thresholding $\left(5\sum I^+ - \sum I^-\right)$. (Note that we have previously fully-differential single-shot dot-product engines are already demonstrated using the same devices in our previous works—see, e.g.,[29,39]), and this simple trick is employed only to enlarge the problem size.

Owing to the single-ended design, we use $g_{ij} = G_{\max}\left(T_{ij}/\max\left(|T_{\max}|\right)\right)$, where $\max\left(|T_{\max}|\right)$ is the maximum absolute weight and $G_{\max}$ is the maximum absolute conductance (40 μS in our experiment). We ground all bitlines (bottom electrode) except the one associated with the selected neuron, which is virtually grounded, and its current is sensed using a B1530A fast measurement unit and a B1500A parameter analyzer. We apply neuron voltages to the switch matrix, connected to both 20 rows and 20 columns of the crossbar. We link top electrodes to the input neurons and bottom electrodes to the output neurons through an E5250A switch matrix.

The eFlash chip, fabricated in Global Foundries 55 nm LPe process, includes a 12×10 redesigned industry-grade split-gate memory array. The packaged chip is previously used for developing a high-performance dot-product engine[52]. Agilent B1500A and B1530A tools are used for measurements and pulse generation. We have developed a custom-made switch matrix on a printed circuit board controlled via a lightweight microprocessor to interface Agilent tools with the chip. More details on the experimental setup, programming, eraser, redesigned layout structure, half-select disturbance immunity, retention, and endurance characteristics are available in Ref.[52]. All eFlash memories are programmed to their targeted states at $V_{WL} = 1.5\,V, V_{CG} = 2.5\,V, V_{BL} = 1\,V, V_{SL} = 0\,V,$ and $V_{EG} = 0\,V$ and operated at the same biasing condition. Further, the devices are tuned one at a time by progressively increasing voltage pulses and using the write-verify algorithm. We have discussed the details of pulse amplitudes and durations in the programming phase in Ref.[52].

As discussed in the main text, weight annealing is implemented by increasing the V_{WL} from 0.7 to 1.5 V linearly and exponentially, which would exponentially and superexponentially increase the synaptic weights, respectively, since devices are operated in weak inversion (see Supplementary Materials S.7). Similar to the memristor-based circuit, we use the single device per synapse topology and compute each update in two cycles. The weights are mapped from software to hardware by using $I_{ij}^T = I_{\max}\frac{T_{ij}}{|T_{\max}|}$ and $I_j^b = I_{\max}\frac{T_j^b}{|T_{max}^b|}$ in which $I_{\max} = 1\mu A$, $T_{\max} = 2$ is the maximum absolute synaptic weight, and $T_{max}^b = 3.694$ is the maximum absolute bias.

Data availability

The data that support the plots within this paper and are available from the corresponding author upon reasonable request.

Received: 23 March 2020; Accepted: 17 November 2020
Published online: 12 August 2021

References

1. Wen, U., Lan, K. & Shih, H. A review of Hopfield neural networks for solving mathematical programming problems. *Eur. J. Oper. Res.* **198**, 675–687 (2009).
2. Cook, W., Lovász, L. & Seymour, P. D. (eds.) *Combinatorial Optimization: Papers from the DIMACS Special Year*, Vol. 20 (American Mathematical Society, 1995).
3. Korte, B. H. *et al. Combinatorial Optimization* Vol. 1 (Springer, 2011).
4. Horio, Y., Ikeguchi, T. & Aihara, K. A mixed analog/digital chaotic neuro-computer system for quadratic assignment problems. *Neural Netw.* **18**, 505–513 (2005).
5. Yamaoka, M. *et al.* A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing. *IEEE J. Solid State Circuits* **51**, 303–309 (2015).
6. Boixo, S. *et al.* Evidence for quantum annealing with more than one hundred qubits. *Nat. Phys.* **10**, 218 (2014).
7. Johnson, M. W. *et al.* Quantum annealing with manufactured spins. *Nature* **473**, 194 (2011).
8. Yamaoka, M., *et al.* 24.3 20k-spin Ising chip for combinational optimization problem with CMOS annealing. In *Proceedings of ISSCC'15*, (San Francisco, CA, 2015).
9. Takemoto, T., *et al.* 2.6 A 2 × 30k-spin multichip scalable annealing processor based on a processing-in-memory approach for solving large-scale combinatorial optimization problems. In *Proceedings of ISSCC'19* (San Francisco, CA, 2019).
10. Sutton, B., Camsari, K. Y., Behin-Aein, B. & Datta, S. Intrinsic optimization using stochastic nanomagnets. *Nat. Sci. Rep.* **7**, 44370 (2017).
11. Inagaki, T. *et al.* Large-scale Ising spin network based on degenerate optical parametric oscillators. *Nat. Photon.* **10**, 415 (2016).
12. Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci.* **81**, 3088–3092 (1984).
13. Tank, D. & Hopfield, J. J. Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Trans. Circuits Syst.* **33**, 533–541 (1986).
14. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.* **79**, 2554–2558 (1982).
15. Joya, G. M., Atencia, M. A. & Sandoval, D. F. Hopfield neural networks for optimization: Study of the different dynamics. *Neurocomputing* **43**, 219–237 (2002).
16. Kirkpatrick, S. Optimization by simulated annealing: Quantitative studies. *J. Stat. Phys.* **34**, 975–986 (1984).
17. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).
18. Chen, L. & Aihara, K. Chaotic simulated annealing by a neural network model with transient chaos. *Neural Netw.* **8**, 915–930 (1995).
19. Akiyama, Y., *et al.* Combinatorial optimization with Gaussian machines. In *Proceedings IEEE International Joint Conference on Neural Networks*, Vol. 1 (1989).
20. Chen, L. & Aihara, K. Chaos and asymptotical stability in discrete-time neural networks. *Phys. D* **104**, 286–325 (1997).
21. Elidan, G., Ninio, M., Friedman, N. & Schuurmans, D. Data perturbation for escaping local maxima in learning. In *AAAI/IAAI* 132–139 (2002).
22. Ninio, M. & Schneider, J. J. Weight annealing. *Phys. A Stat. Mech. Appl.* **349**, 649–666 (2005).
23. Loh, K., Golden, B. & Wasil, E. Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Comput. Oper. Res.* **35**, 2283–2291 (2008).
24. Loh, K. H., Golden, B. & Wasil, E. *A Weight Annealing Algorithm for Solving Two-dimensional Bin Packing Problems, Operations Research and Cyber-Infrastructure* 121–146 (Springer, 2009).
25. Charon, I. & Hudry, O. The noising method: A new method for combinatorial optimization. *Oper. Res. Lett.* **14**, 133–137 (1993).
26. Coy, S. P., Golden, B. L. & Wasil, E. A. A computational study of smoothing heuristics for the traveling salesman problem. *Eur. J. Oper. Res.* **124**, 15–27 (2000).
27. Gu, J. & Huang, X. Efficient local search with search space smoothing: A case study of the traveling salesman problem (TSP). *IEEE Trans. Syst. Man Cybern.* **24**, 728–735 (1994).
28. Coy, S. P., Golden, B. L., Runger, G. C. & Wasil, E. A. Using experimental design to find effective parameter settings for heuristics. *J. Heuristics* **7**, 77–97 (2001).
29. Prezioso, M. *et al.* Spike-timing-dependent plasticity learning of coincidence detection with passively integrated memristive circuits. *Nat. Commun.* **9**, 5311 (2018).
30. Bavandpour, M., *et al.* Mixed-signal neuromorphic inference accelerators: recent results and future prospects. In *Proceedings of International Electron Devices Meeting (IEDM)* (San Francisco, CA, 2018).
31. Mahmoodi, M. R. & Strukov, D. B. An ultra-low energy internally analog, externally digital vector-matrix multiplier based on NOR flash memory technology. In *Proceedings of Design Automation Conference (DAC)* (San Francisco, CA, 2018).
32. Guo, X., *et al.* Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology. In *Proceedings of International Electron Devices Meeting (IEDM)* (San Francisco, CA, 2017).
33. Rajendran, B. & Alibart, F. Neuromorphic computing based on emerging memory technologies. *IEEE J. Emerg. Select. Top. Circ. Syst.* **6**, 198–211 (2016).
34. Burr, G. W. *et al.* Neuromorphic computing using non-volatile memory. *Adv. Phys. X* **2**, 89–124 (2017).
35. Kuzum, D., Yu, S. & Wong, H. P. Synaptic electronics: Materials, devices and applications. *Nat. Nanotechnol.* **24**, 3001 (2013).
36. Kim, H., Nili, H., Mahmoodi, M. R. & Strukov, D. B. 4K-memristor analog-grade passive crossbar circuit. arXiv:1906.12045 (2019).
37. Mahmoodi, M. R. & Strukov, D. B. An ultra-low energy internally analog, externally digital vector–matrix multiplier based on NOR flash memory technology. In *Proceedings of Design Automation Conference (DAC)* (San Francisco, CA, 2018).
38. Guo, X. *et al.* Modeling and experimental demonstration of a Hopfield network analog-to-digital converter with hybrid CMOS/memristor circuits. *Front. Neurosci.* **9**, 488 (2015).
39. Hu, S. G. *et al.* Associative memory realized by a reconfigurable memristive Hopfield neural network. *Nat. Commun.* **6**, 7522 (2015).
40. Kumar, S., Strachan, J. P. & Williams, R. S. Chaotic dynamics in nanoscale NbO_2 Mott memristors for analogue computing. *Nature* **548**, 318 (2017).
41. Mahmoodi, M. R., Prezioso, M. & Strukov, D. B. Versatile stochastic dot-product circuits based on non-volatile memories for high performance neurocomputing and neural optimization. *Nat. Commun.* **10**, 5113 (2019).
42. Mahmoodi, M. R., *et al.* An analog neuro-optimizer with adaptable annealing based on 64×64 0t1r crossbar circuit. In *Proceedings of IEEE International Electron Devices Meeting (IEDM)* 14.7.1–14.7.4 (San Francisco, CA, USA, 2019).

43. Borders, W. A. *et al.* Integer factorization using stochastic magnetic tunnel junctions. *Nature* **573**, 390–393 (2019).
44. Roy, K., Sengupta, A. & Shim, Y. Perspective: Stochastic magnetic devices for cognitive computing. *J. Appl. Phys.* **123**, 210901 (2018).
45. Fukami, S. & Ohno, H. Perspective: Spintronic synapse for artificial neural network. *J. Appl. Phys.* **124**, 151904 (2018).
46. Danial, L. *et al.* Two-terminal floating-gate transistors with a low-power memristive operation mode for analogue neuromorphic computing. *Nat. Electron.* **2**, 596–605 (2019).
47. Assaad, R. S. & Silva-Martinez, J. The recycling folded cascode: A general enhancement of the folded cascode amplifier. *IEEE J. Solid State Circ.* **44**, 2535–2542 (2009).
48. Razavi, B. The StrongARM latch [a circuit for all seasons]. *IEEE Solid State Circuits Mag.* **7**, 12–17 (2015).
49. Bayat, F. M. *et al.* Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. *Nat. Commun.* **9**(1), 2331 (2018).
50. Mahmoodi, M. R., Nili, H., & Strukov, D. B. RX-PUF: Low power, dense, reliable, and resilient physically unclonable functions based on analog passive RRAM crossbar arrays. In *Proceedings of VLSITEC H'18* (Honolulu, HI, 2018).
51. Nili, H. *et al.* Hardware-intrinsic security primitives enabled by analogue state and nonlinear conductance variations in integrated memristors. *Nat. Electron.* **1**(3), 197 (2018).
52. Guo, X., *et al.* Temperature-insensitive analog vector-by-matrix multiplier based on 55 nm NOR flash memory cells. In *Proceedings of CICC'17* (2017).
53. Ushijima-Mwesigwa, H., Negre, C. & Mniszewski, S. M. Graph partitioning using quantum annealing on the d-wave system. In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing* (ACM, 2017).
54. Alibart, F. *et al.* High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. *Nanotechnology* **23**, 075201 (2012).

Acknowledgements

This work was supported in part by a Semiconductor Research Corporation (SRC) funded JUMP CRISP center, NSF/SRC E2CDA grant 1740352, and partially supported by DENSO CORPORATION.

Author contributions

M.R.M., Z. F., and D.B.S. conceived the original concept. M.R.M. developed the experiment and performed the measurements. Z.F. performed the simulations. H.N. fabricated the memristor crossbar. M.R.M. wrote the manuscript. All authors discussed the results.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-020-78944-5.

Correspondence and requests for materials should be addressed to Z.F. or M.R.M.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Combinatorial Optimization by Weight Annealing in Memristive Hopfield Networks Supplementary Materials

Z. Fahimi[1*#], M. R. Mahmoodi[1*#], H. Nili[1], Valentin Polishchuk[2], D. B. Strukov[1]
[1] UC Santa Barbara, Santa Barbara, CA 93106-9560, U.S.A. [2] Linkoping University, 60174 Norrkoping, Sweden
* Equal Contribution [#]{z.fahimi,mrmahmoodi}@ucsb.edu

Overview

Here, we provide supplementary details in support of the main paper. Section 1 briefly discusses the discrete-time Hopfield networks and annealing techniques. Section 2 formulates the optimization problems considered in this study. Section 3 expands upon the details of the 7-node graph partitioning problem. Section 4 and 5 are devoted to the additional graph partitioning, vertex cover, maximum clique, and maximum-independent set simulation results. Sections 6 and 7 include the relevant details of experiments performed with the two representative memory technologies. Section 8 and 9 discuss the performance prospects and the experimental setups, respectively.

1. Hopfield Neural Networks and Annealing Techniques

The focus of this paper is on discrete-time Hopfield neural networks (HNNs) [1,2]. The following Lyapunov energy function describes the dynamics of the network described by the update rule in Eq. 1 of the main text:

$$E = \frac{-1}{2}\sum_{j=1}^{N}\sum_{i=1,\neq j}^{N} T_{ij}U_i(t)U_j(t) - \sum_{j=1}^{N} T_j^b U_j(t). \tag{S1}$$

In the case of symmetric connections ($T_{ij} = T_{ji}$), zero self-feedback weights ($T_{ii} = 0$), and single neuron update at a time, the network always converges to a stable equilibrium point due to its gradient decedent dynamics. The change in the state of j^{th} neuron at epoch t+1 results in ΔE given by

$$\Delta E = -\left(\sum_{i=1,\neq j}^{N} T_{ij}U_i(t) + T_j^{\text{b}}\right)\Delta U_j. \tag{S2}$$

Since ΔU_j and $\sum_{i=1,\neq j}^{N} T_{ij}U_i(t) + T_j^b$ have similar signs, $\Delta E \leq 0$, and since E is bounded, the equilibrium is a stable point. The search space is the interior of the N-dimensional hypercube, with minima located in the corners of it [3,4]. The most critical shortcoming of Hopfield networks (similar to the Ising model and other greedy and local search methods) is the presence of (many) local minima in their energy function. Although Eq. S2 guarantees the monotonic descendence of Lyapunov energy function, a local minima state may easily

trap the network. Therefore, these models, in general, may not find the global optimum solution. However, in many applications, finding an approximately "good" (and sometimes more energy efficient) solution in a reasonable time is preferred to finding the best solution slowly.

2. Combinatorial Optimization with Hopfield Networks

To implement a typical problem within the Hopfield model, we define an (intuitive) energy function, which its minima locate at the same points as the minima of the cost function (of the problem), and map the solution to the neuron states. The energy function is then compared with the generic Lyapunov function of HNN to find the synaptic weights. The inference involves initializing the neuron states or solution and then allowing the recurrent network to converge to a stable state that is interpreted as the ultimate solution. Several well-known combinatorial optimization problems are considered in our study as follows:

2a. Graph Partitioning

Graph bisection, the problem of minimizing the cutsize when partitioning a graph into two sections of nearly equal weight, finds applications in distributed computing and digital VLSI design flow. We define it over an n-vertex undirected graph $G = (V, E)$ in which w_i and e_{ij} $(1 \leq i, j \leq n)$ are the weights of the i^{th} vertex and the edge between adjacent nodes i^{th} and j^{th}, respectively. The objective is to bisect the graph such that the sum of weight vertices assigned to each partition is equal while the sum of the disconnected edge weights is minimum. A single neuron (U_j) is attributed to each vertex (V_j). $U_j = 1$ determines that V_j belongs to partition 1 and $U_j = 0$ is otherwise. The cost function consists of two terms and is given by

$$E = \alpha \sum_{i=1}^{n} \sum_{j=1}^{n} e_{ij}(U_i + U_j - 2U_iU_j) + \sum_{i=1}^{n} \sum_{j=1}^{n} w_iw_j(1 - U_i - U_j + 2U_iU_j)$$

in which $\alpha = 0.5$ is a constant representing the relative importance of these two terms [5]. The first term is considered to minimize the weighted sum of the edges (which belong to each cut), and the second one to balance the sum of the node weights in two partitions. Rearranging and dropping the constant terms leads to

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1,\neq i}^{n} \left(2e_{ij} - 4w_iw_j\right)U_iU_j - \sum_{i=1}^{n} U_i(2w_i \sum_{j=1}^{n} w_j - 2{w_i}^2 - \sum_{j=1}^{n} e_{ij})$$

which is very similar to the general energy function of Hopfield networks (Eq.S1) that is exploited to derive the weights as follows:

$$T_{ij} = 2e_{ij} - 4w_i w_j, \qquad T_i^{\mathrm{b}} = 2w_i \sum_{j=1}^{n} w_j - 2{w_i}^2 - \sum_{j=1}^{n} e_{ij}.$$

Note that -2w_i^2 term in bias weights is missing in Ref. [5].

2b. Minimum Weighted Vertex Cover Problem

The vertex cover is a fixed-parameter tractable and a central problem in parameterized complexity theory and one of 21 NP-complete problems in Karp's seminal work [6]. The adjacency matrix of G, $A = [a_{n\times n}]$, is defined by $a_{ij} = 1$ if there is an edge between i^{th} and j^{th} nodes of graph G, otherwise, $a_{ij} = 0$. The vertex cover C is a subset of G if all the edges of G are adjacent to at least one vertex in the set C. In minimum weighted vertex cover problems, the goal is to find the vertex cover with minimum cardinality (or size) whose total weight is also minimum [7]. Similar to the graph partitioning, one neuron is assigned per vertex. $U_i = 1$ determines that V_j is a part of the vertex cover. The cost function is defined by

$$E = \alpha \sum_{i=1}^{n} w_i U_i + \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(1 - U_i)(1 - U_j)$$

in which α is a constant representing the relative importance of these two terms. The first term is used to minimize the cardinality and total weight, and the second term ensures full coverage of the solution by penalizing the energy whenever there is an edge with endpoints not included in the cover. By dropping the constant terms and comparing them with the generalized energy function, we obtain the weights as

$$T_{ij} = -2a_{ij}, \qquad T_i^{\mathrm{b}} = 2\sum_{j=1}^{n} a_{ij} - \alpha w_i.$$

2c. Maximum Weight Independent Set Problem

In this problem, we are interested in finding the independent subset C of G with maximum total weight, which neither of its nodes is adjacent. The energy function is formulated by

$$E = -\alpha \sum_{i=1}^{n} w_i U_i + \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} U_i U_j,$$

and all the parameters have the same meaning as before. The first term favors larger sets with larger weights and the second term penalizes the energy if there are adjacent nodes in C. Comparing it with the generic energy function leads to

$$T_{ij} = -2a_{ij}, \qquad T_i^{\mathrm{b}} = \alpha w_i.$$

2d. Maximum-Weight Clique Problem

A clique is a subset of G in which all nodes are adjacent to each other, and finding a clique with maximum cardinality and the maximum total weight is an NP-compete problem [6]. It finds many applications, e.g., in bioinformatics and analysis of random processes. We consider $U_i = 1$ when i^{th} node is a part of the maximum clique solution, and use the intuitive energy function

$$E = -\alpha \sum_{i=1}^{n} w_i U_i + \sum_{i=1}^{n} \sum_{j=1}^{n} (1 - a_{ij}) U_i U_j$$

in which the first term ensures that the larger the sum of weights in the maximum clique, the lower the energy, and the second term penalizes the energy when the solution includes non-adjacent nodes. We compare it with the general Lyapunov energy function of the Hopfield model to find the weights:

$$T_{ij} = 2(a_{ij} - 1), \qquad T_i^{\mathrm{b}} = \alpha w_i.$$

3. 7-node graph partitioning problem parameters

An arbitrary graph G=(V, E) with the following elements are used (all numbers are rounded to 2 decimal points for clarity):

$$V = \begin{pmatrix} 0 & 84.17 & 3.88 & 50.64 & 27.53 & 30.86 & 13.21 \\ 84.17 & 0 & 16.31 & 81.04 & 65.84 & 45.65 & 14.21 \\ 3.88 & 16.31 & 0 & 94.10 & 92.18 & 61.67 & 9.09 \\ 50.64 & 81.04 & 94.10 & 0 & 47.69 & 69.23 & 65.51 \\ 27.53 & 65.84 & 92.18 & 47.69 & 0 & 38.24 & 28.12 \\ 30.86 & 45.65 & 61.67 & 69.23 & 38.24 & 0 & 82.18 \\ 13.21 & 14.21 & 9.09 & 65.51 & 28.12 & 82.18 & 0 \end{pmatrix}, E = \begin{pmatrix} 5.64 \\ 3.80 \\ 4.79 \\ 3.42 \\ 8.43 \\ 9.65 \\ 5.01 \end{pmatrix}$$

Then, we employ the energy equations to find the weight matrices:

$$T = \begin{pmatrix} 0 & -100.33 & 23.96 & -100.33 & -135.30 & -156.13 & -86.65 \\ 82.58 & 0 & -40.19 & 110.00 & 3.44 & -55.44 & -47.74 \\ -100.33 & -40.19 & 0 & 122.55 & 22.71 & -61.63 & -77.83 \\ 23.96 & 110.00 & 122.55 & 0 & -20.24 & 6.15 & 62.34 \\ -135.30 & 3.44 & 22.71 & -20.24 & 0 & -249.28 & -112.86 \\ -156.13 & -55.44 & -61.63 & 6.15 & -249.28 & 0 & -29.15 \\ -86.65 & -47.74 & -77.83 & 62.34 & -112.86 & -29.15 & 0 \end{pmatrix}, T^{\mathrm{b}} = \begin{pmatrix} 185.93 \\ -26.32 \\ 67.35 \\ -152.39 \\ 245.76 \\ 272.75 \\ 145.95 \end{pmatrix}$$

The runtime change in synaptic weights is shown in Fig. S1a. The weights are slowly deformed, and the network tends to remain in the transitory ground state during the runtime. The results in Fig. 1Sb compare exponential weight annealing with the baseline on all possible runs. The annealing schedule is $\tau = 40$.

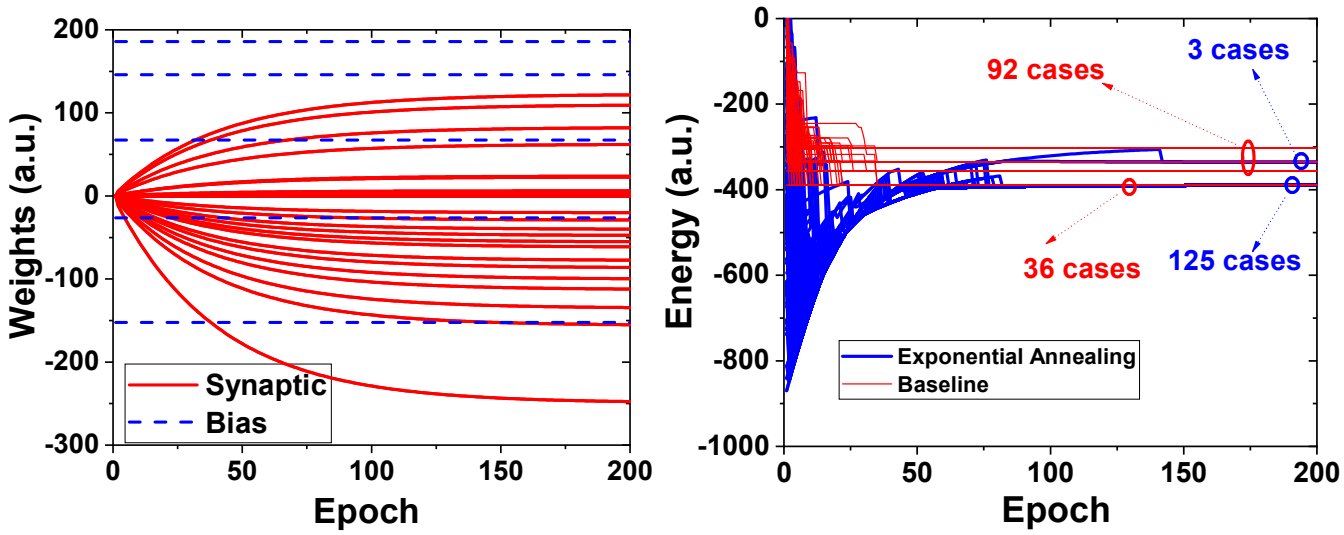


Fig. 1S: (a) The change of synaptic weights for the weight annealing technique when solving the 7-node graph partitioning problem. (b) Comparison between the results obtained by exponential weight annealing versus baseline for all 128 runs.

4. Additional data on graph partitioning

We have conducted three sets of extensive simulations on weighted graph partitioning. For all configurations, the vertex and edge weights are selected randomly in the ranges of [2:*N*] and [0:20], respectively. The impact of scaling the problem size is studied for a fixed number of epochs in the main text (Fig. 1d). Here, the simulation results are provided for the case when we scale up the number of epochs with the problem size. N_{EP} is exponentially increased with respect to the problem size in Fig. 2S. The solution quality of weight annealing

is better than simulated annealing for N>10, as shown in Fig. 2S. Table I shows the parameters used in Figs. 1d, 1e, and 1f (of the main text), and Fig. 2S of supplementary materials.

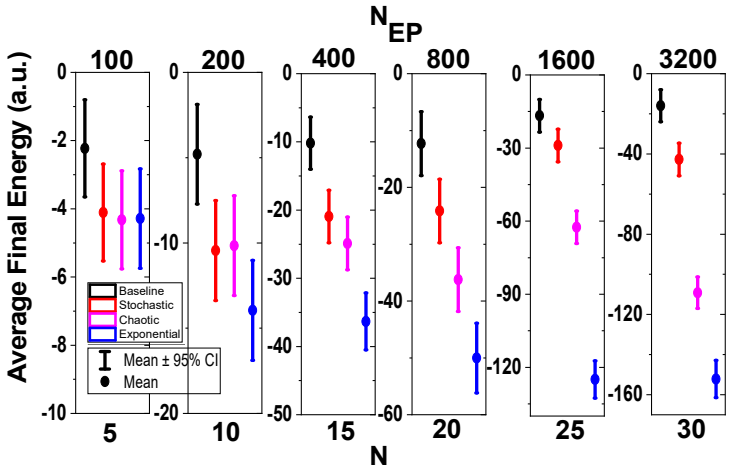


Fig. 2S: Extended graph partitioning simulations: The result of increasing the computational time (N_{EP}) with respect to the graph size (N). Note that, in this figure, the average energy is added by a constant for better clarity.

Table I: The parameters used in the graph-partitioning simulations

	N	Epoch Number	Case numbers	Annealing Schedule		
				Stochastic	Chaotic	Exponential
Fig.1d	5	300	32	50	50	60
	10	300	1000	60	500	60
	15	300	10000	70	1e3	60
	20	300	10000	70	5e3	60
	25	300	10000	100	9e3	60
	30	300	10000	300	2e4	60
	40	300	10000	500	1e5	60
	50	300	10000	1000	2e5	60
Fig.1f	25	100	10000	150	1e3	20
	25	200	10000	200	4e3	40
	25	400	10000	1e3	1e4	80
	25	800	10000	5e3	1e5	160
	25	1600	10000	1e4	1e6	320
	25	3200	10000	5e4	1e7	640
	25	6400	10000	1e5	1e8	1280
	25	12800	10000	1e6	1e9	2560
Fig.2S	5	100	32	30	30	20
	10	200	1000	60	60	40
	15	400	10000	90	3e3	80
	20	800	10000	200	7e3	160
	25	1600	10000	700	1e5	320

	30	3200	10000	2e3	1e6	640
	40	6400	10000	1e4	1e7	1280
	50	12800	10000	1e5	1e8	2560

5. Simulation results on minimum weighted vertex cover, maximum weighted clique, and maximum weighted independent set problems

Similar sets of simulations are performed for three other combinatorial optimization problems as well. Note that simulation parameters (N and N_{EP}) are the same as those for the graph partitioning problem (Table. I), while the annealing schedule might be slightly different since they are manually optimized for each problem type/size. Figs. 3S, 4S, and 5S show the results of the maximum independent set, maximum clique, and minimum vertex cover problems. Similar trends are observed, and weight annealing outperforms other annealing techniques. For the vertex cover, we notice that the weight annealing fails to find the exact solution (i.e., the global optimum) for few large graph configurations. Note that weight annealing is still better or on par with chaotic annealing on average and outperforms simulated annealing in terms of response quality (Top-5 and average energy). We believe that some spurious states generated during the annealing are responsible for those rare underperforming cases.

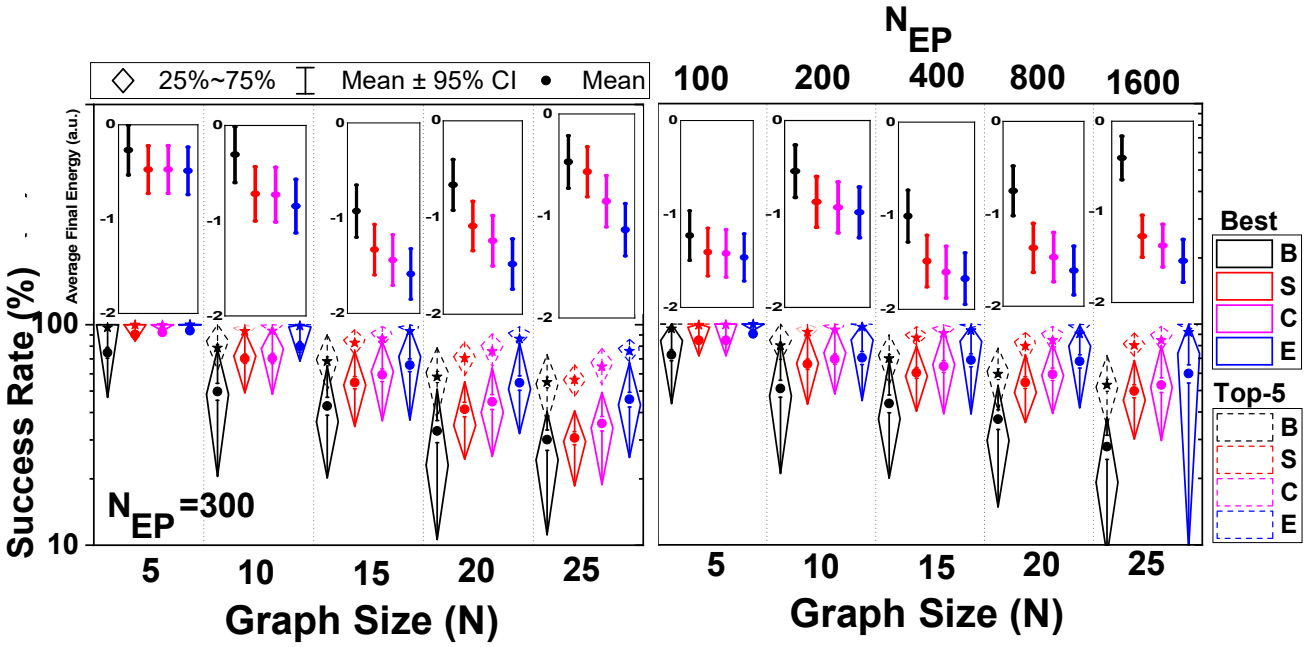


Fig. 3S: Simulation results on 200 random configurations of the maximum weighted independent set problem with various sizes: (a) the success rate and (the average final energy in the inset) for different sizes and fixed computational time (N_{EP}=300), (b) the success rate and (the average final energy in the inset) for different sizes with a varying number of epochs. Note that, in each figure, the actual average energy is added by a constant for better clarity.

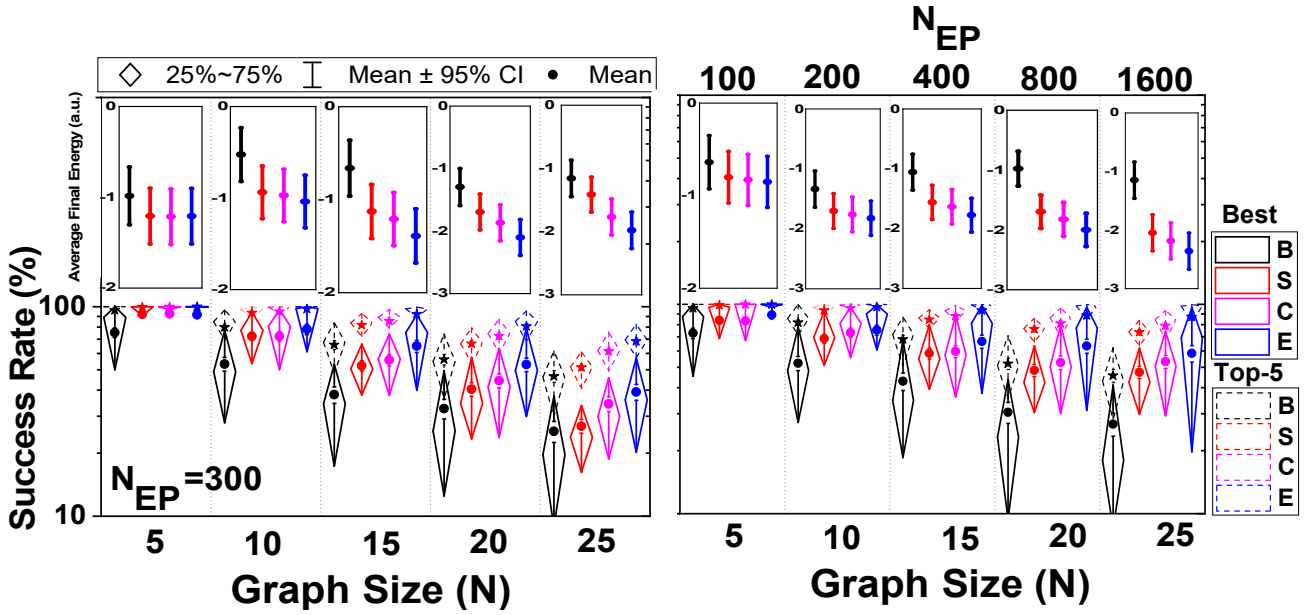


Fig. 4S: Simulation results on 200 random configurations of the maximum weighted clique problem with various sizes: (a) the success rate and (the average final energy in the inset) for different sizes and fixed computational time (N_{EP}=300), (b) the success rate and (the average final energy in the inset) for different sizes with a varying number of epochs. Note that, in each figure, the actual average energy is added by a constant for better clarity.

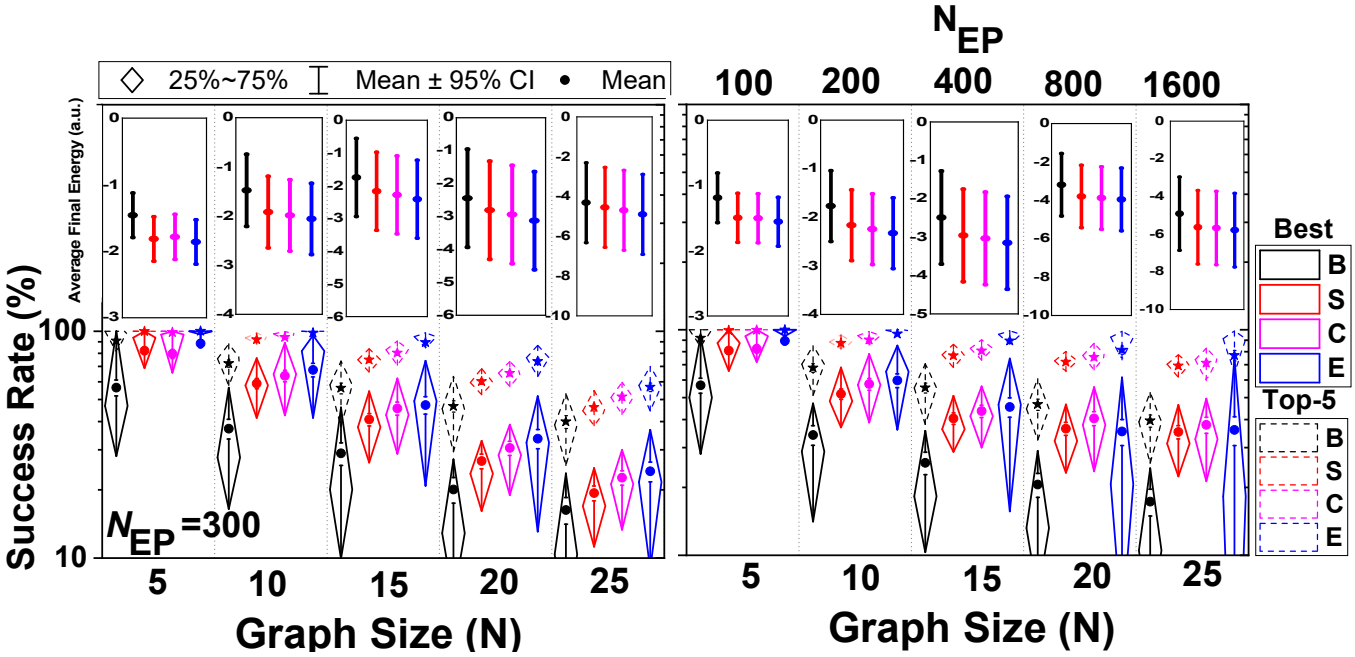


Fig. 5S: Simulation results on 200 random configurations of the minimum weighted vertex cover with various sizes: (a) the success rate and (the average final energy in the inset) for different sizes and fixed computational time (N_{EP}=300), (b) the success rate and (the average final energy in the inset) for different sizes with a varying number of epochs. Note that, in each figure, the actual average energy is added by a constant for better clarity.

6. Additional details on the 13-node graph partitioning experiment

A random graph G=(V, E) with the following analog weights are used in the memristor-based hardware implementations of weight annealing:

$$V=\begin{pmatrix} 0 & 11 & 6 & 4 & 6 & 1 & 1 & 16 & 18 & 25 & 9 & 4 & 5 \\ 11 & 0 & 12 & 29 & 30 & 2 & 16 & 10 & 14 & 5 & 14 & 9 & 17 \\ 6 & 12 & 0 & 30 & 15 & 3 & 25 & 6 & 18 & 0 & 1 & 5 & 14 \\ 4 & 29 & 30 & 0 & 14 & 8 & 3 & 12 & 8 & 3 & 2 & 19 & 5 \\ 6 & 30 & 15 & 14 & 0 & 7 & 16 & 18 & 7 & 8 & 5 & 13 & 5 \\ 1 & 2 & 3 & 8 & 7 & 0 & 5 & 14 & 1 & 28 & 16 & 30 & 7 \\ 1 & 16 & 25 & 3 & 16 & 5 & 0 & 15 & 14 & 5 & 3 & 4 & 0 \\ 16 & 10 & 6 & 12 & 18 & 14 & 14 & 0 & 5 & 13 & 18 & 20 & 13 \\ 18 & 14 & 18 & 8 & 7 & 1 & 14 & 5 & 0 & 16 & 13 & 19 & 14 \\ 25 & 5 & 0 & 3 & 8 & 28 & 5 & 13 & 16 & 0 & 18 & 18 & 17 \\ 9 & 14 & 1 & 2 & 5 & 16 & 3 & 18 & 13 & 18 & 0 & 10 & 2 \\ 4 & 9 & 5 & 19 & 13 & 30 & 4 & 20 & 19 & 18 & 10 & 0 & 14 \\ 5 & 17 & 14 & 5 & 5 & 7 & 0 & 13 & 14 & 17 & 2 & 14 & 0 \end{pmatrix}, E=\begin{pmatrix} 10 \\ 5 \\ 4 \\ 6 \\ 3 \\ 6 \\ 5 \\ 13 \\ 5 \\ 14 \\ 10 \\ 5 \end{pmatrix}.$$

Then, the procedure described in section 2 is utilized to find the synaptic weights and bias matrices of the HNN model:

$$T=-\begin{pmatrix} 0 & 178 & 148 & 192 & 228 & 118 & 238 & 168 & 484 & 150 & 542 & 392 & 190 \\ 178 & 0 & 56 & 42 & 42 & 56 & 88 & 80 & 232 & 90 & 252 & 182 & 66 \\ 148 & 56 & 0 & 20 & 20 & 42 & 46 & 68 & 172 & 80 & 222 & 150 & 52 \\ 192 & 42 & 20 & 0 & 92 & 44 & 114 & 76 & 244 & 94 & 276 & 162 & 90 \\ 228 & 60 & 66 & 92 & 0 & 58 & 112 & 84 & 298 & 104 & 326 & 214 & 110 \\ 118 & 56 & 42 & 44 & 58 & 0 & 62 & 32 & 154 & 4 & 136 & 60 & 46 \\ 238 & 88 & 46 & 114 & 112 & 62 & 0 & 90 & 284 & 110 & 330 & 232 & 120 \\ 168 & 80 & 68 & 76 & 84 & 32 & 90 & 0 & 250 & 74 & 244 & 160 & 74 \\ 484 & 232 & 172 & 244 & 298 & 154 & 284 & 250 & 0 & 228 & 702 & 482 & 232 \\ 150 & 90 & 80 & 94 & 104 & 4 & 110 & 74 & 228 & 0 & 244 & 164 & 66 \\ 542 & 252 & 222 & 276 & 326 & 136 & 330 & 244 & 702 & 244 & 0 & 540 & 276 \\ 392 & 182 & 150 & 162 & 214 & 60 & 232 & 160 & 482 & 164 & 540 & 0 & 172 \\ 190 & 66 & 52 & 90 & 110 & 46 & 120 & 74 & 232 & 66 & 276 & 172 & 0 \end{pmatrix}, T^b=\begin{pmatrix} 1514 \\ 691 \\ 561 \\ 723 \\ 876 \\ 406 \\ 913 \\ 700 \\ 1881 \\ 704 \\ 2045 \\ 1455 \\ 747 \end{pmatrix}.$$

To realize weight annealing within our memristive crossbars, we encode the normalized weights ($T_{ij}<0$ and $T_j^b>0$) to device conductances by using $g_{ij}=G_{\max}(T_{ij}/|W^{\max}|)$ and $g_j^b=G_{\max}(T_j^b/|W^{\max}|)$ where $|W^{\max}|=\max\{|T|,|T^b|\}$, where $G_{\max}$ is a predetermined value. Memristive crossbars inherently implement the dot-product using Ohm and Kirchhoff's laws, i.e.,

$$V(t+1)=\mathrm{f}\left(\sum_{i=1}^{N} g_{ij}V_i(t)+g_j^b V^{\mathrm{ap}}\right). \tag{S3}$$

Note that f(.) is the binary activation function realized by a comparator in peripheral circuits. In our experimental setup, straightforward peripheral functionalities such as current sensing and binary activation

function are emulated by Agilent characterization tools. In exponential weight annealing, $V_i(t) = V_i^{\mathrm{ap}}(t)(1 - \mathrm{e}^{\frac{-t}{\tau}})$ is the applied voltage to the i^{th} input and $V_i^{\mathrm{ap}}(t)$ is either 0 or V_{ap}, based on the neuron state. Given that, we rewrite Eq. S3 as

$$V(t+1) = \mathrm{f}\left(\frac{V^{\mathrm{ap}} G_{\max}}{|W^{\max}|}\sum_{i=1}^{N} T_{ij}(V_i(t)/V^{\mathrm{ap}}) + T_j^{\mathrm{b}}\right)$$

$$= \mathrm{f}\left(\frac{V^{\mathrm{ap}} G_{\max}}{|W^{\max}|}\sum_{i=1}^{N} T_{ij}(V_i^{\mathrm{ap}}(t)(1 - e^{\frac{-t}{\tau}})/V^{\mathrm{ap}}) + T_j^{b}\right)$$

$$= \mathrm{f}\left(\frac{V^{\mathrm{ap}} G_{\max}}{|W^{\max}|}\sum_{i=1}^{N} w_{ij}(t)(V_i^{ap}(t)/V^{\mathrm{ap}}) + T_j^{b}\right), \quad \text{(S4)}$$

which is essentially the hardware implementation of Eq. 1 of the main text.

7. Additional details the 7-node maximum weight independent set experiment

For the eFlash-based experimental demonstration, a graph with the following adjacency matrix and nodal weights are assumed:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}, E = \begin{pmatrix} 6.40 \\ 7.38 \\ 5.05 \\ 1.21 \\ 3.43 \\ 2.02 \\ 6.09 \end{pmatrix}$$

Section S2 discusses the equations for solving the 7-node maximum weight independent set problem. Accordingly, the synaptic weights are found by assuming $\alpha = 0.5$:

$$T = -\begin{pmatrix} 0 & 2 & 2 & 0 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 0 & 2 & 2 & 0 & 0 \\ 0 & 2 & 2 & 0 & 2 & 2 & 2 \\ 2 & 2 & 2 & 0 & 0 & 2 & 2 \\ 2 & 2 & 0 & 2 & 2 & 0 & 2 \\ 2 & 2 & 0 & 2 & 2 & 2 & 0 \end{pmatrix}, T^b = \begin{pmatrix} 3.20 \\ 3.69 \\ 2.52 \\ 0.60 \\ 1.71 \\ 1.01 \\ 3.04 \end{pmatrix}$$

The embedded flash memory array consists of supercells (Fig. 6S), and we can tune each floating-gate cell to a desired current value at nominal biasing conditions. Neglecting the drain-source voltage, $I_{\mathrm{DS}} \approx$

$I_0 e^{\frac{V_{\mathrm{WL}}-V_{\mathrm{th}}}{nV_{\mathrm{th}}}} \approx \theta_0 I_0 e^{\frac{V_{\mathrm{WL}}}{nV_{\mathrm{th}}}}$ where θ_0 is the effective weight of the devices, and other parameters have their usual meaning. We define $I_{ij}(V_{\mathrm{WL}}) = \theta_{ij} I_0 e^{\frac{V_{WL}}{nV_{\mathrm{th}}}}$ and $I_j^b = \theta_j^b I_0 e^{\frac{V_{\mathrm{ap}}}{nV_{\mathrm{th}}}}$. During tunning, we make sure that $I_{ij}(V^{\mathrm{ap}}) = \frac{I_{\max} T_{ij}}{|W^{\max}|}$ and $I_j^b = \frac{I_{\max} T_j^b}{|W^{\max}|}$ where $|W^{\max}| = \max\{|T|,|T^b|\}$, and $I_{\max}$ is a predetermined value. Hence,

$$\theta_{ij} = \frac{I_{\max} T_{ij}}{|W^{\max}| I_0} e^{\frac{-V^{\mathrm{ap}}}{nV_{\mathrm{th}}}}, \qquad \theta_j^b = \frac{I_{\max} T_j^b}{|W^{\max}| I_0} e^{\frac{-V^{\mathrm{ap}}}{nV_{\mathrm{th}}}} \tag{S5}$$

The update rule for the j^{th} neuron is given by

$$V(t+1) = \mathrm{f}\left(\sum_{i=1}^{N} I_{ij}(V_i(t)) + I_j^b\right) \tag{S6}$$

and $V_i(t)$ is the gate-applied voltage to the i^{th} input terminal at iteration t. It is effortless to show that S6 performs the dot-product of (two-state) input voltage vector and (analog) weights. Note that we apply the digital inputs ($V_{\mathrm{WL}} = 0$ V corresponds to ~0 synaptic current and $V_{\mathrm{WL}} = V^{\mathrm{ap}} = 1.5\ V$ corresponds to the 'on' current for the baseline operation) in the gate-line direction and read the summed currents from the bitlines, (e.g., using a transimpedance amplifier or a current conveyor) to $V_{BL} = 1\ V$.

For the exponential weight annealing, we substitute I_{ij} and I_j^{b} in S6 using S5 and obtain

$$V(t+1) = \mathrm{f}\left(\sum_{i=1}^{N} \frac{I_{\max} T_{ij}}{|W^{\max}| I_0} e^{\frac{-V^{\mathrm{ap}}}{nV_{\mathrm{th}}}} I_0 e^{\frac{V_i(t)}{nV_{\mathrm{th}}}} + \frac{I_{\max} T_j^{\mathrm{b}}}{|W^{\max}| I_0} e^{\frac{-V^{\mathrm{ap}}}{nV_{\mathrm{th}}}} I_0 e^{\frac{V_{\mathrm{ap}}}{nV_{\mathrm{th}}}}\right) =$$

$$= \mathrm{f}\left(\frac{I_{\max}}{|W^{\max}|} \sum_{i=1}^{N} T_{ij} e^{\frac{V_i(t)-V^{\mathrm{ap}}}{nV_{\mathrm{th}}}} + T_j^{\mathrm{b}}\right)$$

and $V_i(t) = \frac{V_i^{\mathrm{ap}}(t) t}{\tau}$ (τ is the annealing schedule) for the semi-exponential learning (it is "semi" since the devices have nonlinearities in their *I-V*) and $V_i(t) = V_i^{\mathrm{ap}}(t)(1 - e^{\frac{-t}{\tau}})$ for the super-exponential learning. Fig. 7Sa shows the weight change for the former case, i.e., linear voltage scaling with $\tau = 500$, and the main synaptic weights grow semi-exponentially. Also, since the currents are extremely small for $V_{WL} < 0.7$, the

ground state of the system remains constant for a significant period. We prevent this by starting from $V_{\mathrm{WL},0} = 0.7$ and govern the annealing by

$$V_i(t) = \frac{(V_i^{ap}(t) - V_{WL,0})t}{\tau} + V_{WL,0}.$$

Fig. 7Sb shows the experimental results on the impact of the annealing schedule and linear versus exponential voltage scaling. We obtain a better solution with slower annealing, particularly with the linear scaling in which the trend is much more apparent. The slowest linear scaling annealing, i.e., when $\tau_{\mathrm{lin}} = 500$ leads to the best solution.

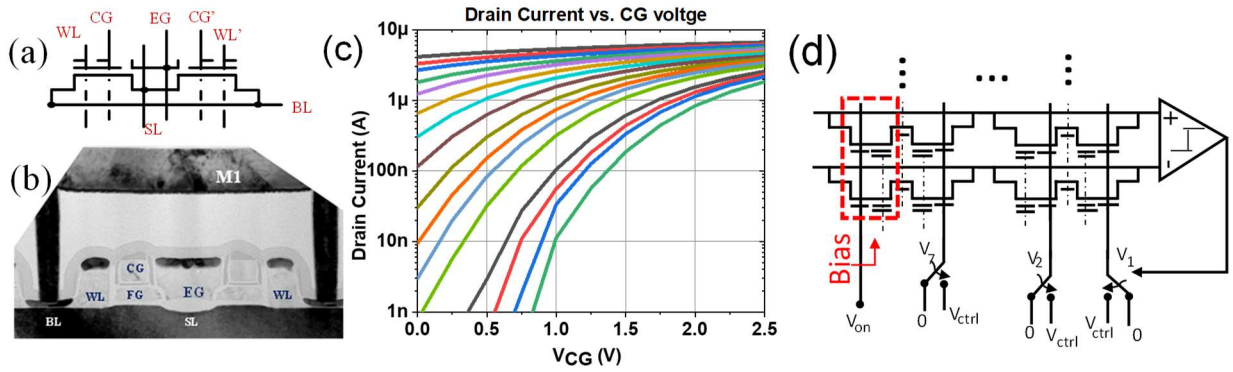


Fig. 6S: (a) The schematic of SST's eFlash supercell consisting of 2 analog-grade floating-gate cells and (b) its TEM image. (c) The weak inversion *I-V* characteristics of a flash cell programmed with 1% accuracy. (d) The schematic of a single channel recurrent neuro-optimizer. For clarity, we show the feedback signal only from the first channel.

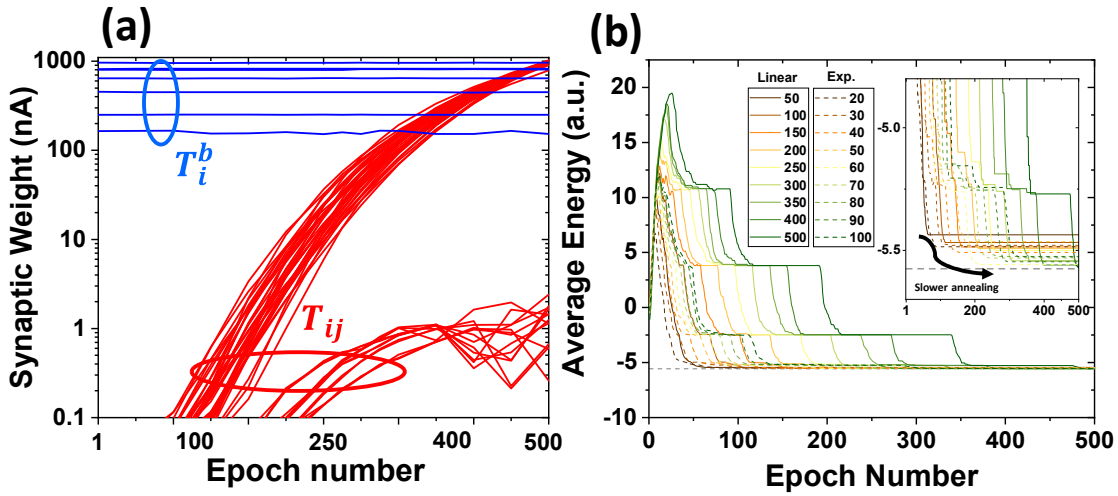


Fig. 7S: Extended experimental results of the 7-node maximum weighted independent set problem: (a) weight evolution during weight annealing (linear voltage scaling), (b) the impact of changing the annealing schedule (τ) for both cases of linear and exponential voltage scaling.

8. The performance of the proposed circuit

We design the peripheral circuits (amplifiers, current reference, and dynamic comparator) in Global Foundries 55 nm CMOS technology to determine the power, area, and speed of the circuit. 3-stage recycling folded cascode amplifiers are considered for designing the transimpedance amplifiers in the presynaptic and postsynaptic circuits and use the strongARM latch to develop the dynamic comparator. Flash memories have a large output impedance in weak inversion. This allows us to use faster and more compact circuits, e.g., current conveyors, for designing the readout circuit (see [25]). We assume 130×64 arrays, which can implement a full-differential 64×64 Hopfield network. In our estimation, we also consider the area of tuning circuits, analog switches, and decoders and exclude clocking and IO circuits as they can be shared with other system-on-chip modules. The circuits are designed targeting 2 ns settling time in dot-product operation and 0.5 ns sampling time in the comparator (hence, ~2.5 ns/update) in the typical corner. Such design parameters lead to ~0.33 pJ/update, ~3,038 μm^2 active area with 4 F^2 memristors, assuming 1 µs annealing schedule (2,000 epochs per run). For the eFlash technology, we obtain 0.09 pJ/update and ~10,992 μm^2 active area with ~110 F^2 redesigned eFlash memories, assuming 1 µs annealing schedule. These preliminary data suggest that our approach could potentially be $10^2\times$ faster and $10^5\times$ more energy efficient as compared to the most efficient conventional methods based on graphics processor units on the same task.

9. Experimental Setups

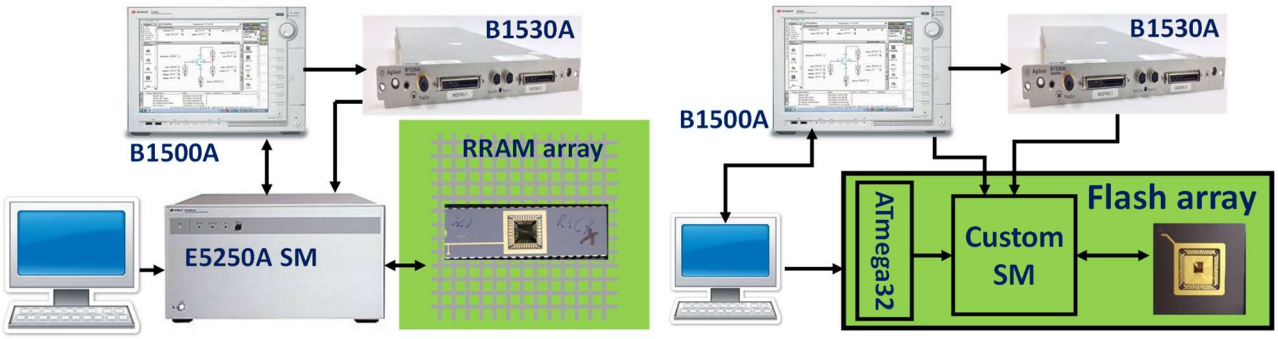


Fig. 8S: The Experimental setup for (a) the 13-node graph partitioning implemented with memristive devices and (b) the 7-node maximum weighted independent set realized with an eFlash memory array. The experimental setups comprise of a personal computer to control parameter analyzer B1500A, arbitrary waveform generator B1530A, and a switch matrix (a low-leakage Agilent E5250A in (a) and a custom circuit for (b)). Both memristive crossbar and flash array are mounted on custom host circuit board and connected to their corresponding switch matrices.

References

[1] Hopfield, John J., and David W. Tank. ""Neural" computation of decisions in optimization problems." Biological cybernetics 52.3 (1985): 141-152.

[2] Hopfield, John J. "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the national academy of sciences 79.8 (1982): 2554-2558

[3] Tank, Df, and J. J. Hopfield. "Simple' neural 'optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit." IEEE transactions on circuits and systems 33.5 (1986): 533-541.

[4] Joya, Gonzalo, M. A. Atencia, and Francisco Sandoval. "Hopfield neural networks for optimization: study of the different dynamics." Neurocomputing 43.1-4 (2002): 219-237.

[5] J. Ramanujam and P. Sadayappan, "Mapping combinatorial optimization problems onto neural networks", *Information Science*, vol. 82, pp. 239-255, 1995.

[6] R. M. Karp, "Reducibility among combinatorial problems", *Complexity of computer computations, Springer*, Boston, MA, pp. 85-103, 1972.

[7] M. N. Syed, and P. M. Pardalos, "Neural network models in combinatorial optimization", *Handbook of Combinatorial Optimization*, pp. 2027-2093, 2013.

[8] M. R. Mahmoodi, and D. Strukov. "An ultra-low energy internally analog, externally digital vector-matrix multiplier based on NOR flash memory technology." *Proceedings of the 55th Annual Design Automation Conference*. 2018.