Solving mazes with memristors: A massively parallel approach

Yuriy V. Pershin[1,*] and Massimiliano Di Ventra[2,†]

[1]*Department of Physics and Astronomy, USC Nanocenter, University of South Carolina, Columbia, South Carolina 29208, USA*

[2]*Department of Physics, University of California San Diego, La Jolla, California 92093-0319, USA*

(Received 18 March 2011; revised manuscript received 11 July 2011; published 14 October 2011)

Solving mazes is not just a fun pastime: They are prototype models in several areas of science and technology. However, when maze complexity increases, their solution becomes cumbersome and very time consuming. Here, we show that a network of memristors—resistors with memory—can solve such a nontrivial problem quite easily. In particular, maze solving by the network of memristors occurs in a massively parallel fashion since all memristors in the network participate simultaneously in the calculation. The result of the calculation is then recorded into the memristors' states and can be used and/or recovered at a later time. Furthermore, the network of memristors finds all possible solutions in multiple-solution mazes and sorts out the solution paths according to their length. Our results demonstrate not only the application of memristive networks to the field of massively parallel computing, but also an algorithm to solve mazes, which could find applications in different fields.

DOI: 10.1103/PhysRevE.84.046703 PACS number(s): 02.70.−c, 87.18.Sn, 73.50.Fq, 73.63.−b

I. INTRODUCTION

Mazes are a class of graphical puzzles in which, given an entrance point, one has to find the exit via an intricate succession of paths, with the majority leading to a dead end, and only one, or a few, correctly solving the puzzle. Mazes—sometimes also called labyrinths—have been known since ancient times, the oldest presumably being the one created by Daedalus in Crete, as passed on by Greek mythology a few thousand years ago. They are used as prototype models in graph theory, topology, robotics, traffic optimization, psychology, and in many other areas of science and technology [1–8]. The ability to solve a maze is particularly important for transportation and robot control systems, e.g., to find the shortest path from a given point to another. In neuroscience, different maze solving algorithms could be compared with the approaches used by humans to solve mazes. In this way, we could better understand the functioning of the human brain and advance new modeling of neural processes.

Algorithms to solve mazes vary from the simplest—and extremely slow—random mouse to mathematical search algorithms that operate on a sequential fashion to find the exit. However, all these methods suffer from very slow solution times when the complexity of the maze increases, with solution times sometimes increasing dramatically with increasing local connectivity of the network. Moreover, some researchers have demonstrated that certain biological and chemical systems can solve mazes [4,9]. For instance, in Ref. [4], an unexpected behavior of a primitive organism was revealed by showing that an amoeba finds the minimum-length path between two points in a labyrinth connecting separate food sources. Such methods, however, are also slow and do not seem to be suitable for large mazes with complex connectivity.

In this paper, we suggest and demonstrate a different strategy for solving mazes that, instead, is based on massively parallel computation as afforded by a network of memristors (short for memory resistors) [10]. Memristors are resistors whose resistance depends on the history of the system and, therefore, can remember their past dynamics. These systems, which belong to the larger class of memory circuit elements—that also includes memcapacitors and meminductors [11]—are attracting considerable attention due to their usefulness in diverse research areas [12], ranging from memories [13,14] to neuromorphic computing and learning [15–17].

A network of memristors, complemented by some other standard electronic elements (such as field-effect transistors), forms a *memristor processor* that, at the present level of technology, can be fabricated experimentally. In this paper, instead, we use numerical simulations to explicitly show that a memristor processor is able to solve mazes. Most importantly, unlike existing approaches, the memristor processor requires only one step to find the maze solution. We emphasize though that the type of massive parallelism used in the memristive processor can be thought of as *analog parallelism*. It is essentially different from that used in conventional computers and rather shows some similarity with massively parallel computing with organic layers demonstrated recently [18].

It is also worth noting here that we could perform maze solving with a network of meminductors or memcapacitors. We choose to work with memristors since they are the most studied so far and can easily be realized experimentally, thus, allowing a practical and straightforward implementation of our algorithm.

The remainder of this paper is organized as follows. In Sec. II, we introduce the main part of our proposal, namely, the memristive processor. We discuss all important operation details of the memristive processor including its initialization, maze mapping, and finally, finding the maze solution. Section III illustrates the dynamics of maze solving by the memristive processor. In this section, we first introduce a mathematical model of the unit part of the processor, the memristor. Numerical simulations of solutions of both single-path and multiple-path mazes by the memristive processor are presented. It is important to note that, in the case of multiple-path mazes, the memristive processor not only finds all solutions, but also sorts them out according to their length. We conclude with some final remarks and considerations on the proposed approach.

*pershin@physics.sc.edu

†diventra@physics.ucsd.edu

II. MEMRISTIVE PROCESSOR

Let us then start by mapping a given maze into a network of memristors. This is shown in Fig. 1. First of all, we superimpose periodical arrays of vertical and horizontal lines on the maze. The period of this array corresponds to the intrinsic period of the maze (for nonperiodic mazes, the period of line arrays should be selected in such a way to take all important maze features into account). The crossing points of vertical and horizontal lines define grid points of the memristive network. The network consists of basic units (memristors plus switches) connecting grid points. Since the direction of current flow in the network is not known *a priori*, the polarity of adjacent memristors (indicated by the black thick line in the memristor symbol in Fig. 1) is chosen to be alternating. It is assumed that external signals can be applied to any grid points for the purpose of initializing the memristors' states as well as to read the calculation results. The externally controlled switches are used to define the topology of the maze: A maze wall is modeled by a switch in the not-connected state. Such an architecture allows modeling different mazes on the same memristive processor without the need to fabricate a specific processor for each maze.

The processor initialization can be performed by simultaneous application of GND, V_1 voltages in a chessboardlike pattern to all grid points of the memristive network [see Fig. 2(a)]. The calculation performed by this memristive processor occurs when a constant voltage V is applied across the two grid points corresponding to the entrance and exit points of the maze as shown in Fig. 1. In this case, the current flows only along those memristors that connect the entrance and exit points. The state of these memristors is changed by the current, thus, the maze is solved in a massively parallel way, since all memristors in the network participate simultaneously in the calculation. Specifically, assuming that, by the initial moment of time, all memristors were initialized in the high-resistance (OFF) state, as time passes, every other memristor along the solution path changes its resistance, eventually switching into the low-resistance (ON) state. Therefore, the chain of memristors in the ON state (or in an intermediate state if memristors did not have enough time to completely switch to the ON state) connecting the entrance and exit points represents the maze's solution. A possible approach to read the calculation result is described in Fig. 2(b).

Furthermore, the states of the memristors connecting the entrance and exit points represent a solution at any given finite time after the initial one. If there are multiple paths, then the shortest one would contain less memristors and, thus, offers less resistance than the longest one, with all intermediate paths (in terms of length) offering a proportionate resistance. Therefore, since current flows in inverse proportion to the resistance of a path, at any given time, the change of state of a given path is proportional to the current in the path. The different paths of the maze then can be identified by the different state their memristors have during (or after) the switching process. In the next section, we explicitly discuss this solution feature. Here, we only point out that the memristive processor can be reduced to a network of standard resistors and, in principle, this could also solve a maze. However, in this case, additional external memory and local voltage

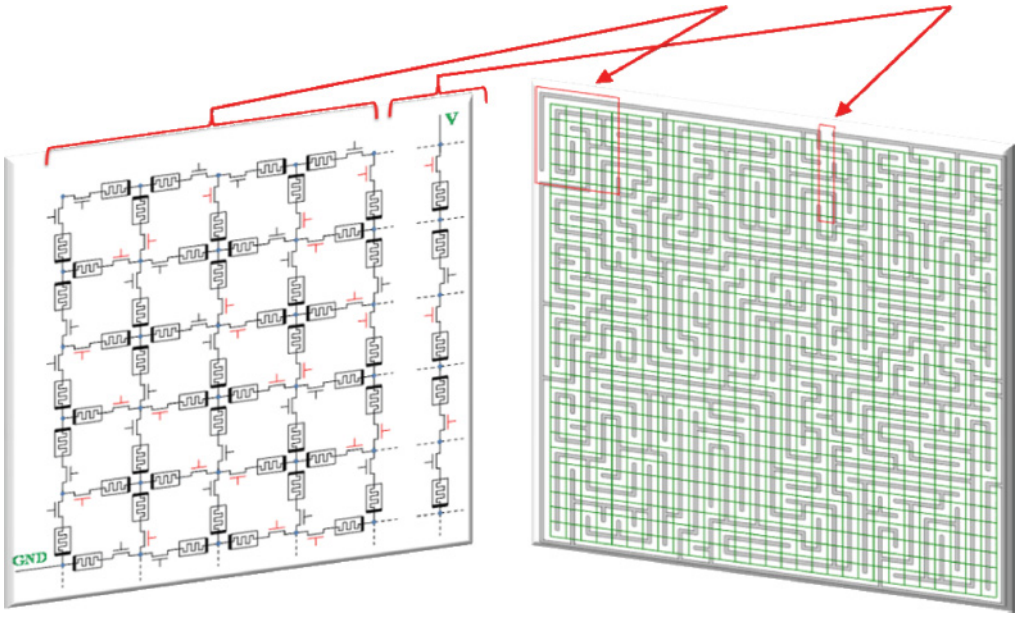

FIG. 1. (Color online) Maze mapping into a network of memristors (the memristive processor). (Right panel) The maze is covered by an array of vertical and horizontal lines having the periodicity of the maze. (Left panel) Architecture of the network of memristors in which each crossing between vertical and horizontal lines in the array (in the right panel) is represented by a grid point to which several basic units consisting of memristors and switches (field-effect transistors) are attached. The maze topology is encoded into the state of the switches such that, if the short line segment connecting neighboring crossing points in the array crosses the maze wall, then the state of the corresponding switch is not connected [shown with red (dark gray) symbols]. All other switches are in the connected state. The external voltage (V) is applied across the connection points corresponding to the entrance (V) and exit [ground (GND)] points of the maze.

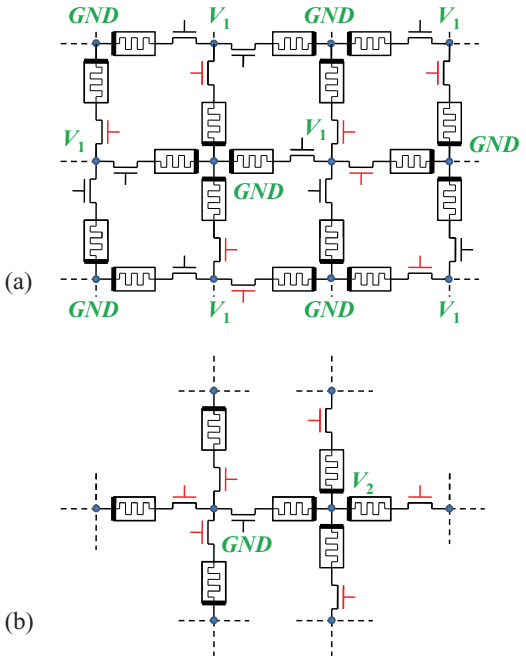

FIG. 2. (Color online) (a) Initialization of the network of memristors can be performed by a simultaneous application of GND and appropriately selected V_1 voltages in a chessboardlike pattern to all grid points for a sufficiently long period of time to securely switch memristors to the OFF state. Note that, since the memristors' polarities are alternated in the network, the corresponding voltage polarities are also alternated. During the initialization, all switches should be in the connected state or should already encode the maze (as shown on the plot). (b) In order to read a memristor's state, such a memristor (shown in the center) can be disconnected from the rest of the circuit setting the corresponding switches to the not-connected state [shown with red (dark gray) symbols] and its own switch to the connected state and can be tested by the application of a short small-amplitude voltage pulse (or double bipolar voltage pulse to minimize the disturbance of the memristor's state) of an amplitude V_2.

measurement hardware is required as the resistive network does not store the calculation results. In addition, in the case of a standard resistive network, previous calculation results cannot be used easily in subsequent calculations. Finally, a network of memristors is a complex nonlinear dynamical system that can potentially be used to solve a larger class of problems, of which optimization ones—as those described in this paper—are only a small class.

Experimentally, the suggested network could be fabricated using, e.g., complementary metal-oxide-semiconductor (CMOS) + molecular-scale devices architecture [19] combining a single memristor layer with a conventional CMOS layer. In the past, it was demonstrated that many different classes of materials and systems exhibit memristive behavior, including binary oxides (TiO_2, CuO, NiO, CoO, Fe_2O_3, MoO, VO_2) [20–25], perovskite-type oxides ($Pr_{1-x}Ca_xMnO_3$, $SrTiO_3$:Cr) [26–30], sulfides (Cu_2S,Ag_2S) [31–33], semiconductors (Si, GaAs, ZnSe-Ge) [34–36], spintronics materials [37–39], and organics [40–42]. Moreover, hybrid memristor-CMOS integrated circuits for reconfigurable logic applications as well as memristive memory chips combining memristive materials with transistors were recently developed experimentally [14,43]. Therefore, fabrication of a memristive processor is possible at the current level of technology. A small-scale version of a memristive processor can be built using memristor emulators [44].

III. NUMERICAL SIMULATIONS

A. Model

In this section, we present numerical simulations of the dynamics of memristive processors. Numerical modeling of memristive networks is easily implemented and, thus, offers by itself, a practical computational algorithm for maze solving. This computational approach, however, requires multiple computational steps (see below) as opposed to the real memristive processor discussed in the previous section, which needs only a single step to perform the whole computation.

For the sake of clarity, we use a simple model of memristor [45], whose memristance (memory resistance) is given by

$$R_{ij}^M = R_{\rm ON} x_{ij} + R_{\rm OFF}(1 - x_{ij}), \tag{1}$$

where $R_{\rm ON}$ and $R_{\rm OFF}$ are minimal and maximal values of memristance, x_{ij} is the dimensionless internal state variable bound to the region $0 \leqslant x_{ij} \leqslant 1$, and (i, j) are grid indexes of a memristor to identify its location in the network. Similar to Ref. [45], we choose the dynamics of x_{ij} to be given by

$$\frac{dx_{ij}}{dt} = \alpha I_{ij}(t), \tag{2}$$

where α is a constant and $I_{ij}(t)$ is the current flowing through the memristor (ij). At each time step, the potential at all grid points is found as a solution of Kirchhoff's current law equations obtained using a sparse matrix technique. The corresponding change in the memristors' states was computed using Eq. (2).

All numerical results reported in this paper were obtained using model parameters $R_{\rm ON} = 10\ \Omega$, $R_{\rm OFF} = 100\ \Omega$, $R_{ij}^M(t = 0) = 91\ \Omega$, $x(t = 0) = 0.1$, and $\alpha = 10^4/(\mathrm{s\,A})$ for all memristors. The applied voltage was selected to be equal to 50 V during the first 0.1 s time interval and -50 V at $t > 0.1$ s. The sign of the applied voltage was changed in order to better represent the maze solution as discussed below. Two types of mazes were considered: a single-path maze and a multiple-(two-) path maze.

B. Results

We have applied the computational scheme described above to several mazes. In all cases, the correct maze solutions were found. Examples of our calculations are shown in Figs. 3 and 4. These mazes have been mapped on a $n \times n$ square memristive processor with $n = 30$. Here, n^2 gives the total number of grid points. It is evident that the square geometry selected for this particular case is not generally required. Mazes of any shape (e.g., rectangular, circular, and irregular) can be mapped on a square memristive processor of sufficient size.

Figure 3 presents results of the solution of a single-path maze (see also Supplementary Movie 1 [46]). We have found that, with these parameters, it takes approximately 0.06 s to

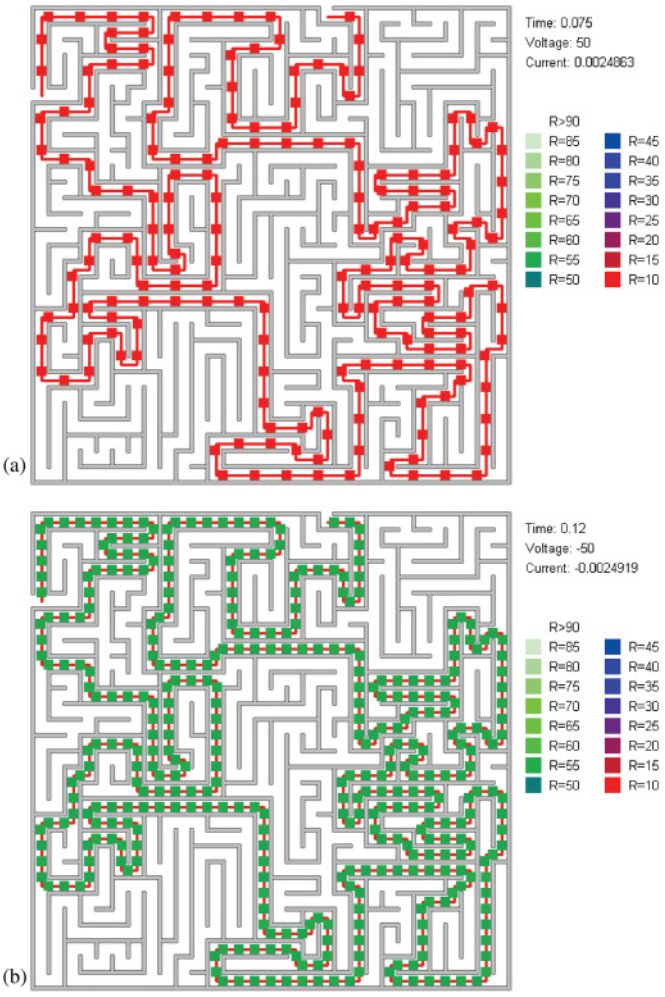

FIG. 3. (Color online) Solution of a single-path maze. (a) Network state at $t = 0.075$ s. The chain of memristors in the low-memristance state [shown by red (dark gray) dots connected by a red (dark gray) line] clearly connects the entrance and exit points of the maze [note that memristors in the OFF state—when $R_{ij}^{M}(t = 0) > 90\ \Omega$—are not shown]. Here, every other memristor along the solution path is in the low-memristance state. (b) Network state at $t = 0.12$ s. Note that, at $t = 0.1$ s, the sign of the applied voltage has been changed. At $t = 0.12$ s, each memristor along the solution path shows the maze solution. The resistance is in ohms, the voltage is in volts, and the current is in amperes.

switch every other memristor along the solution path into the low-resistance ON state. The resulting sequence of memristors in the low-resistance state represents the maze solution as shown in Fig. 3(a). This maze solution can be seen better if we change the sign of the applied voltage and wait some time. Then, the resistance of memristors—along the solution path—that are in the ON state will increase, and the resistance of memristors in the OFF state will decrease. Figure 3(b) captures a specific moment of time when these memristances are equal. At this moment in time, every memristor along the solution path is in the same intermediate state, and thus, the solution of the maze is more visible.

A two-path maze, whose solution is given in Fig. 4, was obtained from the maze shown in Fig. 3 by removing a single segment of the wall (its location is shown by an arrow in Fig. 4).

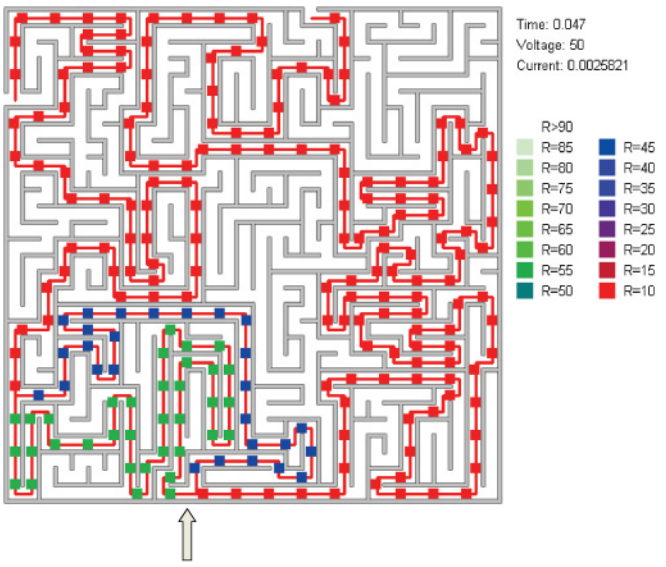

FIG. 4. (Color online) Solution of a multiple-path maze. Network state at $t = 0.047$ s. The maze solution contains two common segments [red (dark gray) dots connected by a red (dark gray) line], and two alternative segments of different lengths close to the left bottom corner. The memristance in the shorter segment [blue (very dark gray) dots connected by a red (dark gray) line] is smaller than that in the longer segment [green (light gray) dots connected by a red (dark gray) line] since the current through the shorter segment is larger and, consequently, the change in the memristors' state along this segment is larger. The arrow at the bottom indicates where a modification of the maze from Fig. 3 has been made (position of the removed segment of the wall). The resistance is in ohms, the voltage is in volts, and the current is in amperes.

In this maze, the current flowing through a common segment [red (dark gray) dots in Fig. 4] splits between two possible paths according to their resistances. Therefore, memristors in the common segment change their resistance faster than those in the two possible paths (see Fig. 4 and Supplementary Movie 2 [46]). Moreover, comparing memristors from two possible paths, memristors in the shorter path change their resistance faster than those in the longer path. As discussed previously, this allows sorting all possible solutions according to their length in such a way that the resistances of the memristors along shorter paths are smaller.

We note that the types of memristors and simulation parameters (such as α and V) have been selected for the sole purpose of demonstration of our idea. The resulting switching time (on the order of 0.05 s, see the above discussion) is much longer than the typical switching times of nanoscale memristive devices that can be as short as 5 ns (see, for example, parameters of nanoionic resistive random access memories in Refs. [12,47]). Since, with an appropriate choice of voltage and current magnitude, the switching time of all the memristors along the solution path is on the order of the switching time of a single memristor, we can argue that the minimum time required to solve a maze of arbitrary complexity by our method can be as short as few nanoseconds or even less, since, as discussed above, full switching is not required to find the solution. Moreover, only one step (a single voltage pulse) is needed to find the solution. Therefore, the approach we

suggest is *a priori* more efficient than any multistep algorithm in present use. In addition, in simulations with memristors described by Eqs. (1) and (2), we always have observed the system evolution toward a unique (for a given maze) stable solution. Therefore, the success rate of correct solution is 100% in our scheme.

Regarding the time complexity of the numerical modeling algorithm, it is mainly determined by the solution of a set of linear Kirchhoff's current law equations. The best theoretical estimate for a linear system (the Coppersmith-Winograd algorithm [48]) is $O(n^{2.376})$. This estimate, thus, provides the theoretical time complexity of the numerical maze solving algorithm suggested in this paper. We note that the time complexity of the breadth-first search algorithm that can be used to find the shortest path is only slightly better. In particular, it is given by $O(V + E) \sim O(n^2)$, where V and E are the number of vertices and edges, respectively [49]. However, we would like to emphasize once more that the *hardware* implementation of the memristive processor requires a single computational step and, thus, outperforms all existing maze solving algorithms.

IV. CONCLUSION

In conclusion, we have shown how a network of memristors—a memristive processor—can solve mazes in a massively parallel way. This approach can be realized experimentally with available systems and devices or simply can be implemented on a computer. The hardware implementation of the memristive processor is superior to any existing maze solving methods, and therefore, it is ideal when the complexity of the maze increases with increasing local connectivity of the graph. Although we have considered a processor based on memristors, a network of memcapacitors or meminductors [11] can also be used for massively parallel calculations. For now, several experimental systems exhibiting memcapacitive and meminductive properties are known [12]. Electronically, memcapacitive and meminductive circuits can be emulated using memristors [50,51].

Moreover, we anticipate that the memristive processor can facilitate the solution of—or can solve—many other computational problems. Examples of such problems include the traveling salesman problem, graph theory problems, etc. The memristive processor then can be used as a complete computational device or as a supplemental tool for traditional computing hardware. Since memristors (as well as memcapacitors and meminductors) are generally asymmetric devices, unidirectional graphs also can easily be realized on appropriate memristive processors. We, thus, envision their use in a large set of applications in both basic science and technology.

ACKNOWLEDGMENT

M.D. acknowledges partial support from the NSF Grant No. DMR-0802830.

[1] S. Pellow, P. Chopin, S. E. File, and M. Briley, J. Neurosci. Methods **14**, 149 (1985).
[2] P. Modesti and A. Sciomachen, Eur. J. Oper. Res. **111**, 495 (1998).
[3] A. Tero, R. Kobayashi, and T. Nakagaki, Physica A **363**, 115 (2006).
[4] T. Nakagaki, H. Yamada, and A. Toth, Nature (London) **407**, 470 (2000).
[5] D. A. Crowe, B. B. Averbeck, M. V. Chafee, J. H. Anderson, and A. P. Georgopoulos, J. Cogn. Neurosci. **12**, 813 (2000).
[6] A. Nelson, E. Grant, J. Galeotti, and S. Rhody, Rob. Aut. Syst. **46**, 159 (2004).
[7] A. Blum, P. Raghavan, and B. Schieber, SIAM J. Comp. **26**, 110 (1997).
[8] P. D. Reiners, *Robots, Mazes, and Subsumption Architecture* in IBM Developer Works (IBM, New York, 2007), available at [http://www.ibm.com/developerworks/java/library/j-robots].
[9] I. Lagzi, S. Soh, P. J. Wesson, K. P. Browne, and B. A. Grzybowski, J. Am. Chem. Soc. **132**, 1198 (2010).
[10] L. O. Chua, IEEE Trans. Circuit Theory **18**, 507 (1971).
[11] M. Di Ventra, Y. V. Pershin, and L. O. Chua, Proc. IEEE **97**, 1717 (2009).
[12] Y. V. Pershin and M. Di Ventra, Adv. Phys. **60**, 145 (2011).
[13] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, IBM J. Res. Dev. **52**, 449 (2008).
[14] S. Dietrich, M. Angerbauer, M. Ivanov, D. Gogl, H. Hoenigschmid, M. Kund, C. Liaw, and M. Markert, IEEE J. Solid-State Circuits **42**, 839 (2007).
[15] Y. V. Pershin, S. La Fontaine, and M. Di Ventra, Phys. Rev. E **80**, 021926 (2009).
[16] Y. V. Pershin and M. Di Ventra, Neural Networks **23**, 881 (2010).
[17] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, Nano Lett. **10**, 1297 (2010).
[18] A. Bandyopadhyay, R. Pati, S. Sahu, F. Peper, and F. Fujita, Nat. Phys. **6**, 369 (2010).
[19] K. K. Likharev and D. B. Strukov, in *Introducing Molecular Electronics*, edited by G. F. G. Cuniberti and K. Richter (Springer, Berlin, 2005), Vol. 657, pp. 447–477.
[20] J. J. Yang, M. D. Pickett, X. Li, D. A. A. Ohlberg, D. R. Stewart, and R. S. Williams, Nat. Nanotechnol. **3**, 429 (2008).
[21] I. H. Inoue, S. Yasuda, H. Akinaga, and H. Takagi, Phys. Rev. B **77**, 035105 (2008).
[22] D. Lee, D.-j. Seong, I. Jo, F. Xiang, R. Dong, S. Oh, and H. Hwang, Appl. Phys. Lett. **90**, 122104 (2007).
[23] S. Seo *et al.*, Appl. Phys. Lett. **85**, 5655 (2004).
[24] T. Driscoll, H.-T. Kim, B.-G. Chae, B.-J. Kim, Y.-W. Lee, N. M. Jokerst, S. Palit, D. R. Smith, M. Di Ventra, and D. N. Basov, Science **325**, 1518 (2009).
[25] T. Driscoll, H.-T. Kim, B. G. Chae, M. Di Ventra, and D. N. Basov, Appl. Phys. Lett. **95**, 043503 (2009).
[26] A. Asamitsu, Y. Tomioka, H. Kuwahara, and Y. Tokura, Nature (London) **388**, 50 (1997).
[27] R. Fors, S. I. Khartsev, and A. M. Grishin, Phys. Rev. B **71**, 045305 (2005).
[28] D. S. Kim, Y. H. Kim, C. E. Lee, and Y. T. Kim, Phys. Rev. B **74**, 174430 (2006).

[29] G. I. Meijer, U. Staub, M. Janousch, S. L. Johnson, B. Delley, and T. Neisius, Phys. Rev. B **72**, 155102 (2005).
[30] Y. B. Nian, J. Strozier, N. J. Wu, X. Chen, and A. Ignatiev, Phys. Rev. Lett. **98**, 146403 (2007).
[31] K. Terabe, T. Hasegawa, T. Nakayama, and M. Aono, Nature (London) **433**, 47 (2005).
[32] T. Tamura, T. Hasegawa, K. Terabe, T. Nakayama, T. Sakamoto, H. Sunamura, H. Kawaura, S. Hosaka, and M. Aono, Jpn. J. Appl. Phys. **45**, L364 (2006).
[33] R. R. Waser and M. Aono, Nat. Nater. **6**, 833 (2007).
[34] S. H. Jo and W. Lu, Nano Lett. **8**, 392 (2008).
[35] Y. Dong, G. Yu, M. C. McAlpine, W. Lu, and C. M. Lieber, Nano Lett. **8**, 386 (2008).
[36] S. H. Jo, K.-H. Kim, and W. Lu, Nano Lett. **9**, 870 (2009).
[37] Y. V. Pershin and M. Di Ventra, Phys. Rev. B **78**, 113309 (2008).
[38] Y. V. Pershin and M. Di Ventra, Phys. Rev. B **79**, 153307 (2009).
[39] X. Wang, Y. Chen, H. Xi, H. Li, and D. Dimitrov, El. Dev. Lett. **30**, 294 (2009).
[40] D. R. Stewart, D. A. A. Ohlberg, P. A. Beck, Y. Chen, R. S. Williams, J. O. Jeppesen, K. A. Nielsen, and J. F. Stoddart, Nano Lett. **4**, 133 (2004).
[41] Y.-S. Lai, C.-H. Tu, D.-L. Kwong, and J. S. Chen, Appl. Phys. Lett. **87**, 122101 (2005).
[42] F. Alibart, S. Pleutin, D. Guerin, C. Novembre, S. Lenfant, K. Lmimouni, C. Gamrat, and D. Vuillaume, Adv. Funct. Mater. **20**, 330 (2010).
[43] Q. Xia *et al*., Nano Lett. **9**, 3640 (2009).
[44] Y. V. Pershin and M. Di Ventra, IEEE Trans. Circ. Syst. I **57**, 1857 (2010).
[45] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, Nature (London) **453**, 80 (2008).
[46] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevE.84.046703 for movies of solution dynamics.
[47] ITRS. The International Technology Roadmap for Semiconductors, 2009 ed. (ITRS, 2009). [http://www.itrs.net].
[48] D. Coppersmith and S. Winograd, J. Symb. Comput. **9**, 251 (1990).
[49] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. (MIT Press, Cambridge, MA, 2009).
[50] Y. V. Pershin and M. Di Ventra, Electron. Lett. **46**, 517 (2010).
[51] Y. V. Pershin and M. Di Ventra, Electron. Lett. **47**, 243 (2011).