Scalable Method to Find the Shortest Path in a Graph with Circuits of Memristors

Alice Mizrahi,[1,2,*] Thomas Marsh,[1] Brian Hoskins,[1] and M. D. Stiles[1]

[1] *National Institute of Standards and Technology, Gaithersburg, Maryland, USA*
[2] *Maryland NanoCenter, University of Maryland, College Park, Maryland, USA*

(Received 12 September 2018; revised manuscript received 24 October 2018; published 14 December 2018)

Finding the shortest path in a graph has applications in a wide range of optimization problems. However, algorithmic methods scale with the size of the graph in terms of time and energy. We propose a method to solve the shortest-path problem using circuits of nanodevices called memristors and validate it on graphs of different sizes and topologies. It is both valid for an experimentally derived memristor model and robust to device variability. The time and energy of the computation scale with the length of the shortest path rather than with the size of the graph, making this method particularly attractive for solving large graphs with small path lengths.

DOI: 10.1103/PhysRevApplied.10.064035

I. INTRODUCTION

Optimization problems, such as sequencing decisions, resource allocation, or navigation, are omnipresent and important. They can be solved by being mapped to finding the shortest path between two nodes in a graph. Algorithms to do this task exist, but the time and energy they consume scale with the size of the graph and thus become prohibitive for large systems [1–7]. In this paper, we propose a method to solve the shortest-path problem that scales consistently better than with the size of the graph. Our approach is inspired by biological systems that need to solve optimization problems to be energy efficient. For example, members of ant colonies work in parallel to find the shortest path to their food without supervision [8]. Algorithms known as ant colony optimization take inspiration from this phenomenon to solve optimization problems in an approximate but efficient way [9–11]. More broadly, the field of swarm intelligence provides optimization algorithms inspired from animal populations [12–14]. However, these algorithms are limited when they run on conventional computers, which compute sequentially, suppressing the parallelism present in the biological phenomena.

Pershin and Di Ventra proposed to solve the shortest-path problem directly in hardware, using nanodevices called memristors, which have dynamics that provide reinforcement mechanisms similar to the ones at play in the ant colony optimization [15–18]. Memristors are defined by having conductances that change when subjected to electrical current [19–21]. As a voltage is applied to a network of memristors forming a graph, more current flows through the shortest branch because that branch has the lowest resistance. This increased flow causes an increase of conductance of the memristors on the shortest path, attracting even more current. When a steady state is reached, the memristors on the shortest path have a much larger conductance than the ones on the longer paths, making it possible to electrically read out the shortest path.

In this work, we build on Pershin and Di Ventra's idea and propose a modified method. The advantage of this method is that it does not require prior knowledge about the shortest path. We show that the time and energy consumed by this method scale with the length of the shortest path rather than with the size of the graph, which makes it potentially more efficient than algorithmic methods.

Sections II and III provide background on memristors and describe how they can be used to find the shortest path in a graph. Sections IV and V present our method and validate it through numerical simulations of large numbers of randomly generated graphs of various sizes and topologies. We show that it is valid for a realistic, experimentally derived memristor model and parameters and that it is robust to device variability. Section VI addresses the key advantage of this method by showing that the time and energy it consumes scale with the length of the shortest path. Finally, Sec. VII addresses the hardware implementation of this method.

II. MEMRISTOR-BASED OPTIMIZATION

Memristors are a class of devices that exhibit hysteretic behavior: their electrical conductance can be modified in

*alice.mizrahi@u-psud.fr

a nonvolatile fashion. The conductance can take values between two extreme states. The device is said to be "*on*" or "*off* " when in its highest or lowest conductance state, respectively. The conductance can be repeatedly increased and decreased by running current through the device [19–21].

A graph can be represented by a circuit of memristors: the nodes of the graph are electrical junctions connected by memristors implementing the edges of the graph. An example is shown in Fig. 1(a): each black dot corresponds to a node and each colored rectangle to a memristor. The goal is to find the shortest path between the start and end nodes, marked by green stars in Fig. 1(a).

A voltage is applied across these nodes, causing current to flow through the circuit. The system is initialized with all memristors in their *off* state. The shortest path, i.e., composed of the lowest number of memristors, is more conductive than the other paths. As a consequence of Kirchhoff laws, more current flows through the memristors on the shortest path than through the other memristors. This causes their conductance to increase, drawing even more current to them and thus creating a reinforcement mechanism. On the other hand, the memristors outside the shortest path have little current flowing through them, so their conductance does not increase. This behavior is shown in Fig. 1(b), which presents the evolution of the conductance of the memristors belonging to the shortest path (in red) and the others (in black) versus time. After some time, the system reaches a steady state where the memristors on the shortest path are *on* while the others are *off*. This state is depicted in Fig. 1(a), where the conductance of each memristor is represented by the color of the rectangle. The successfully found shortest path can be visually observed in red, contrasting with the rest of the graph in purple. As their state is nonvolatile, the voltage across the circuit can be turned off and the individual memristors can be measured in order to determine the shortest path (as described in Sec. IV).

Note that the shortest path is found here without supervision: it emerges from the dynamical evolution of the system. Furthermore, the computation is parallel, with each memristor evolving at the same time. This parallelism is key to how the time and energy consumed by the computation scale as the size of the problem increases, as detailed in Sec. VI.

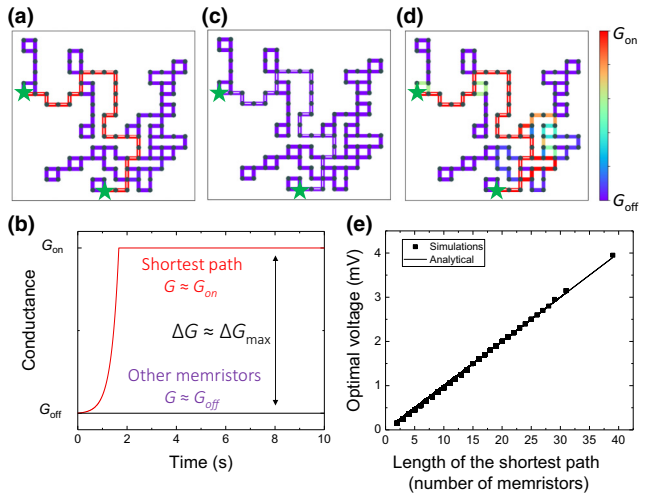

FIG. 1. (a)–(d) Schematic of a memristor circuit at its steady state for different applied voltages: 4 mV for panel (a), 1 mV for panel (c), and 10 mV for panel (d). The colors correspond to the conductance of each memristor. (b) Conductance versus time for the shortest-path memristors (in red) and the longer paths' memristors (in black) for the simulation illustrated in panel (a), with an applied voltage of 4 mV. Note that, for these ideal devices, the curves of all memristors in each path category superimpose. The difference ΔG is indicated by a double arrow. (e) Optimal voltage (leading to the highest ΔG) versus the length of the shortest path, i.e., the number of memristors composing it. Each data point corresponds to a simulation on one of the 1996 randomly generated square grid graphs. The solid line corresponds to the analytical value $N/\gamma\tau G_{on}$.

III. MEMRISTOR MODELS

In this work, we consider both a simple generic memristor model and a more complex realistic memristor model, derived from experiments. We start by studying a simple generic memristor model, where

$$I = V[G_{on}x + G_{off}(1-x)], \tag{1}$$

where I is the current though a memristor and V the voltage across it. $G_{on} = 10^{-1}$ S and $G_{off} = 10^{-4}$ S are the extreme conductance states of the memristors. x is an internal state variable, bounded between 0 and 1, describing the conductance dynamics of the device. The conductance can be increased by running current through the device:

$$\frac{dx}{dt} = \gamma|I| - \frac{x}{\tau}, \tag{2}$$

where $\gamma = 10^6\,\mathrm{A}^{-1}\,\mathrm{s}^{-1}$ and $\tau = 0.1$ s is the decay constant of the device. The decay term shows that the devices are not strictly nonvolatile, but are practically so on useful time scales. We numerically simulate the evolution of circuits of such devices by solving Kirchhoff equations at each junction. Because of the linear relationship between current and voltage in this model, the dynamics of the circuit can be described as a system of linear equations, which allows us to perform a large number of simulations.

In order to validate our results, we also use the memristor model developed by Chang *et al.*, with the device parameters they extracted from experiments [22]. The model corresponds to a $Pd/WO_3/W$ stack. Applying a

voltage across the device induces oxygen vacancy migration in the oxide. The width and length of the created conducting filament determine the conductance of the device, through the internal state variable x:

$$I = (1-x)\alpha[1-\exp(-\beta V)] + x\gamma \sinh(\delta V), \quad (3)$$

$$\frac{dx}{dt} = \lambda\left[\exp\left(\eta_1 V\right) - \exp\left(-\eta_2 V\right)\right] - \frac{x}{\tau}, \quad (4)$$

where $\alpha = 5\times10^{-7}$ S, $\beta = 0.5$ V^{-1}, $\gamma = 4\times10^{-6}$ S, $\delta = 2$ V^{-1}, $\lambda = 4.5$ s^{-1}, $\eta_1 = 0.004$ V^{-1}, $\eta_2 = 4$ V^{-1}, and $\tau = 10$ s are device parameters.

Note that, in this model, the polarity of the voltage matters, as it can grow or shrink the filament. Since most optimization problems map to directed graphs, devices sensitive to the polarity of the voltage could be an asset. For broader applications, schemes where two memristors of opposed polarities are connected in parallel for each edge could be used, as proposed in Ref. [16]. Furthermore, other types of memristors, such as phase-change memories, are not sensitive to the polarity of the voltage across them [23–26]. The nonlinear form of the current-voltage relationship complicates the solution of the system of equations, leading us to use a commercial circuit simulator to solve the system of equations.

IV. OBTAINING THE CORRECT RESULT

We examine the importance of the control voltage, i.e., the voltage applied across the circuit, and show that using a constant voltage is not practical for applications. Panels (a), (c), and (d) of Fig. 1 show the steady state of one single graph after the application of three different voltages, as described in Sec. II. The generic linear memristor model described in Sec. III is used. In Fig. 1(a), where a voltage of 4 mV is applied, the computation is successful. However, Figs. 1(c) and 1(d) show examples of incorrect computation. In Fig. 1(c), the voltage (1 mV) is too low to significantly increase the conductance of any memristor. On the other hand, in Fig. 1(d), the voltage (10 mV) is too high, so the conductance of some memristors on longer paths is increased comparatively to those on the shortest path. Only an optimal range of control voltage leads to a successful computation of the shortest path.

To investigate the role of the control voltage on the computation, we define a metric of success for solving the shortest-path problem. The input of the problem is the topology of the memristor circuit, i.e., the graph. The proposed method can provide the length of the shortest path and identify the shortest path. To read the output of the computation (once a steady state is reached or after the evolution of the system is stopped by the user, as we describe in Sec. V), the control voltage across the circuit is removed.

The length of the shortest path is determined by measuring the conductance of the whole circuit between the start and end nodes. As the conductance of the shortest path dominates the global conductance and as the nominal conductance of the memristors in their *on* state is known, the length of the path can be deduced.

Determining the shortest path requires additional underlying circuitry. The shortest path starts with the start node—where the voltage source is applied—and is constructed node by node. The conductance of each memristor connected to this node is measured. The next node on the shortest path is the one connected to the current node through the highest conductance memristor. This process is repeated until the end node—the node connected the ground—is reached. This method allows the user to read the shortest path by probing only a fraction of the memristors in the network, proportional to the length of the shortest path.

This method requires electrical access to each node of the graph. This can be implemented by building the memristors on top of conventional complementary metal-oxide semiconductor (CMOS) circuitry with mixed analog and digital design. An analog circuit measures the conductances and conventional digital switches give access to the desired nodes.

For this method to read the correct shortest path, it is required that, at each step, the measured memristor belonging to the shortest path have a higher conductance than the others. We thus define as the metric of success, ΔG, the smallest difference in conductance between a memristor on the shortest path and a memristor outside the path but connected to the same node of the path. The shortest-path problem is successfully solved if $\Delta G > 0$. The highest possible success is $\Delta G_{\max} = G_{on} - G_{off}$. The value of ΔG sets how sensitive the measurement circuits need to be. The lower ΔG is, the stricter the requirements for the voltage noise. Devices with a large $G_{on} - G_{off}$ are therefore desirable. In the successful example of Fig. 1(a), $\Delta G \simeq \Delta G_{\max}$, as observed in Fig. 1(b). On the other hand, the results in Figs. 1(c) and 1(d) both exhibit $\Delta G \simeq 0$.

In order to determine the optimal control voltage, we generate thousands of graphs of different sizes, based on a square grid with randomly removed edges, similar to the one shown in Fig. 1. The start and end nodes are selected randomly for each graph. Dead-end branches, which can lead to floating memristors, are removed and so are nodes not connected to the start and end nodes. We only select graphs that have a unique shortest path. We generate graphs using different grid sizes and different probabilities for the edges to be removed. The correct shortest path is determined using the standard breadth-first search algorithm.

For each graph, the corresponding circuit is numerically simulated at various control voltages. The optimal control voltage is the voltage at which ΔG is the highest.

Figure 1(e) shows that the optimal voltage is proportional to the length of the shortest path. In order to turn on a memristor, the time derivative of its x must be non-negative when x approaches 1; i.e., the voltage across the device is greater than $1/\gamma\tau G_{on}$. The lowest control voltage to achieve this is $N/\gamma\tau G_{on}$, where N is the number of memristors on the shortest path. This optimal control voltage will turn on the shortest path but not any longer paths. As observed in Fig. 1(e), this matches simulation results. Our simulations show that using a constant control voltage too far from the optimal control voltage gives results like those in panels (c) and (d) of Fig. 1, making such an approach impractical for applications for which the length of the shortest path is not known in advance.

V. USING A VOLTAGE RAMP

We propose a method that does not require prior knowledge of the shortest path length. We leverage the fact that the shortest path is turned on (i.e., has the conductances of its memristors increase to G_{on}) at lower control voltages than longer paths, which suggest the use of a voltage ramp. Figure 2(a) shows the evolution of ΔG with time as the voltage is increased. We observe a sharp increase in ΔG, which corresponds to the shortest path turning on. The increase in conductance of the memristors on the shortest path creates an increase in the global conductance of the circuit, shown in Fig. 2(b). This increase can be detected by measuring the current going in and out of the circuit. The turning on of the shortest path corresponds to a sharp kink in the current, as shown in Fig. 2(c), and thus a drop below zero in the second time derivative of the current, as shown in Fig. 2(d). When this drop is measured, the control voltage is turned off and the result of the computation is read out as described in Sec. IV. Note that here the evolution of the system is stopped, contrary to the constant control voltage method where a steady state is reached.

Figure 3 presents statistics on ΔG obtained from simulations and shows that this method, using a single voltage ramp, can successfully find the shortest path in thousands of graphs of various sizes. In addition to the square-grid-based topology (blue bars), we perform similar simulations on graphs with a small-world topology (red bars). Small-world networks stand in between regular and random graphs and describe many interesting problems [27]. These networks are discussed in more detail in Sec. VI. Here, we generate small-world networks of different sizes and levels of randomness. For both topologies, all simulations exhibit ΔG well above zero, which validates this method.

We test the validity of our method for realistic devices by performing simulations on a square-grid topology with the realistic memristor model and parameters described in Sec. III [22]. Figure 4 shows that ΔG is more widely

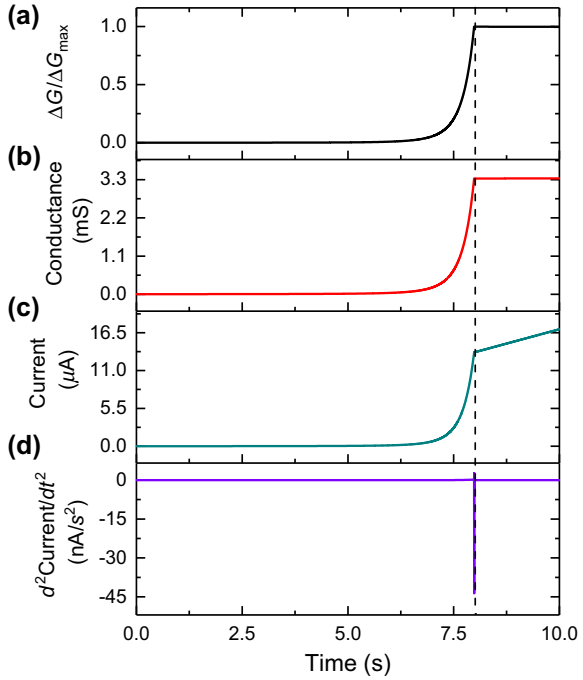

FIG. 2. Evolution with time of (a) the conductance difference $\Delta G/\Delta G_{\max}$, (b) the global conductance of the circuit, (c) the current through the circuit, and (d) the second time derivative of this current. The voltage ramp starts from 0.1 mV and increases at a rate of 0.5 mV/s. The dashed line corresponds to the time of the result detection.

spread than for the generic linear model. This is due to the fact that the realistic model produces smoother behavior and a weaker reinforcement mechanism. However, our approach remains valid, as all graphs exhibit $\Delta G > 0$. Here, the ramp rate of the voltage is chosen to be slow enough to detect the onset of the shortest path even when the latter is just a few memristors long. As higher ramp rates lead to faster computations, the ramp rate could be increased when it is known that the shortest path is longer

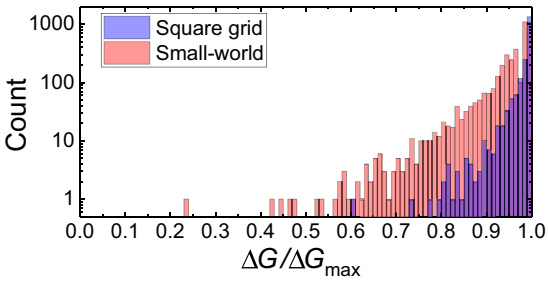

FIG. 3. Histograms of the metric of success $\Delta G/\Delta G_{\max}$ for simulations on randomly generated graphs, using the generic linear model on 1996 square-grid-based graphs (blue bars) and 4797 small-world networks (red bars). The voltage ramp starts from 0.1 mV and increases at a rate of 0.5 mV/s.

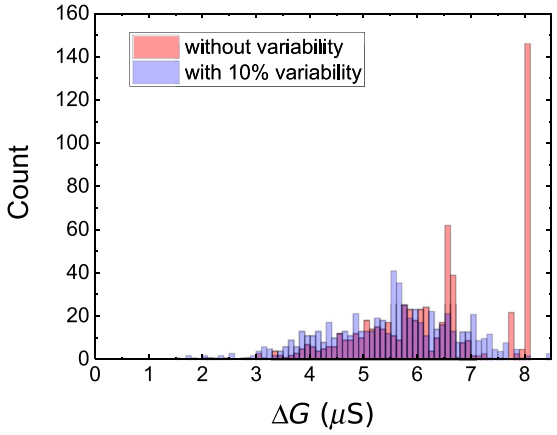

FIG. 4. Histograms of the metric of success ΔG for simulations on 658 randomly generated graphs following the square grid topology, using the experimentally obtained realistic model from Ref. [22] without variability (red bars) and with 10% variability on all device parameters (blue bars). With device variability, each simulation is done on one graph with one set of parameters randomly chosen from a Gaussian distribution around the nominal parameter value and of 10% standard deviation. The set of graphs with and without variability are the same. The voltage ramp starts from 0 V and increases at a rate of 1 mV/s.

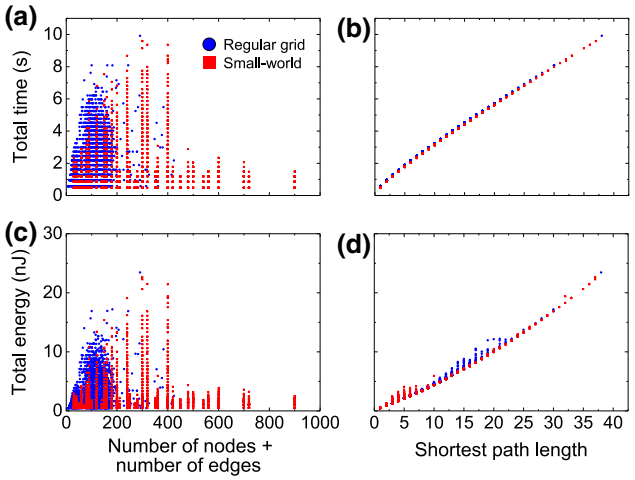

FIG. 5. Total time of the computation versus (a) the size (number of nodes plus number of edges) and (b) the length of the shortest path. Total energy consumed versus (c) the size and (d) the length of the shortest path. Blue circles correspond to square grids and red squares correspond to small-world networks. Each symbol corresponds to one of the 1996 square-grid and 4797 small-world graphs.

than a given value. Materials research could also help building memristors with characteristics leading to higher increase rates.

Furthermore, we investigate the influence of device variability. As shown in Fig. 4, even with 10% variability on all device parameters, our approach still works. For larger graphs and longer paths, we expect the result to become more sensitive to device variability and voltage noise, leading the method to make errors. However, the obtained path has a length that is close to the length of the shortest path. The result is nearly optimal and satisfactory for many problems.

Interestingly, variability can be an asset. For simplicity, in this study, we restrict ourselves to shortest-path problems with a unique solution. However, we observe that, in the case of graphs in which there are two shortest paths of equal length between the considered nodes, systems with ideal devices tend to have lower ΔG than in the case with unique solutions. Qualitatively, the system tries to turn on all shortest paths simultaneously, which prevents the winner-take-all reinforcement mechanism to take place properly. However, device variability makes one of the shortest paths intrinsically more conductive and easier to turn on, which lets the system choose this path over the others and turn it on completely, thus increasing ΔG. This is an interesting example of how, in bio-inspired computing, features of nanodevices usually seen as drawbacks can be beneficial.

VI. SCALING OF THE TIME AND ENERGY CONSUMPTION

In order to evaluate the potential use of our method, we study how the time and energy required by the computation scale with the size of the graph. The energy is estimated as the integral over time of the total current through the circuit times the voltage across the circuit. Additional energy will be spent for detecting the current second time derivative, reading the result, and setting up the circuit, but this is out of the scope of this study as designing the full architecture of the system would be required.

Figures 5(a) and 5(b) present the time and energy consumption versus the number of nodes plus the number of edges for all our simulations with the generic model (corresponding to Fig. 3). This combination is a common way to characterize the size of a graph [1]. We observe no correlation. However, Figs. 5(c) and 5(d) show that the time and energy consumption correlate strongly with the length of the shortest path. Moreover, this scaling does not depend on the topology of the graph.

These results indicate a key advantage of the proposed method. Conventional algorithmic methods typically scale with the number of nodes and the number of edges, because the different nodes and edges are explored sequentially [1–7]. In the present hardware implementation, the current explores the entire circuit in parallel, which makes the time and energy consumption independent of the size of the graph. This method would be particularly efficient for large graphs with small shortest paths. Such graphs

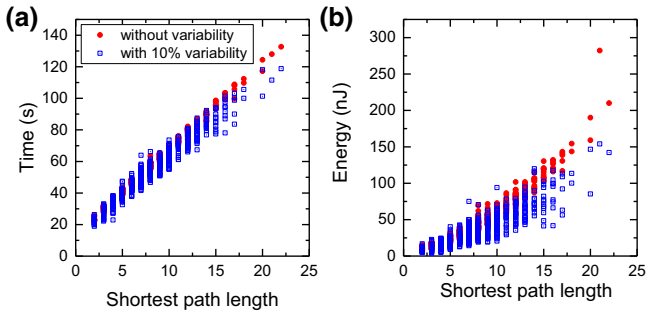

FIG. 6. (a) Total time of the computation versus the length of the shortest path. (b) Total energy consumed versus the length of the shortest path. Each symbol corresponds to a simulation using the realistic memristor model either without variability (red full circles) or with 10% variability on each parameter (blue squares), on one of the 658 square grid graphs. With device variability, each simulation is done on one graph with one set of parameters randomly chosen from a Gaussian distribution around the nominal parameter value and of 10% standard deviation. The sets of graphs with and without variability are the same.

include small-world networks. Standing between regularity and randomness, these networks are composed of many short-range connections and a few long-range connections. They exhibit high clustering and low shortest paths [27]. Small-world networks have been shown to describe many systems with important applications, such as power grids, the structure of the web, social media, and neural networks [27–30].

We investigate the effect of a realistic memristor model and device variability by computing the time and energy consumption corresponding to the simulations used in Fig. 4. We observe that the scaling laws stay valid: as shown in Fig. 6, the time and energy consumption depend on the length of the shortest path. The fact that the time and energy consumption appear to scale independently of the topology or memristor model is promising for the ease of material implementation, as many types of devices could be used, and for the breadth of applications, as many types of graphs could be solved.

VII. OUTLOOK FOR A RECONFIGURABLE GRAPH SOLVER

In order for this approach to be of practical use, it must be reconfigurable for different problems. Finding the shortest path between different nodes of the same graph simply requires connecting the voltage source and ground to the new nodes. Modifying the graph is nontrivial. Nearest-neighbor connections could be implemented by CMOS switches that are opened or closed to form the desired graph, as proposed in Ref. [15]. Longer-range connections could be implemented by another set of memristive devices, arranged in conventional crossbar topologies. The hybrid CMOS/molecular (CMOL) architecture, proposed in Ref. [31], has been shown to give high densities of connections, which could be useful here [32,33]. Building one physical system capable of implementing any graph problem is unrealistic. However, it would be possible to have specialized chips for types of graph problems. There would be one underlying hierarchical structure appropriate to the graph type, as well as many reconfigurable connections implementing specific problems. Such hierarchical architectures have been shown to be efficient at simulating large artificial neural networks [34].

VIII. CONCLUSION

We propose a scheme to find the shortest path in a graph problem using a circuit of memristors. It includes procedures to detect and read the result. Our scheme does not require prior knowledge about the shortest path and has been validated on a large number of graphs of various sizes and topologies. We show that this scheme works for realistic device models and is robust to variability in the range of explored graphs. We imagine that, as graphs get larger, the variability needs to be smaller to guarantee that the approach finds the best solution. Otherwise, it might find a nearly optimal solution rather than the best.

Like many bio-inspired computing schemes, the proposed method is most appropriate for applications where energetic rather than precision constraints are strict. It scales with the length of the shortest path in terms of time and energy consumption because of its intrinsic parallelism. This is a key advantage compared to conventional algorithmic methods that scale with the size of the graph. In particular, it is best for studying large graphs with small shortest paths, such as social networks or power grids. In order to confirm the practical usefulness of the method, further research is required to determine the graph size at which the favorable scaling makes this approach better than those running on conventional hardware. However, these results are promising for hardware implementations of systems capable of performing a fast and energy-efficient analysis of large graphs.

More broadly, the field of swarm intelligence is rich and implementing its concepts in hardware offers many paths toward energy-efficient computing. For example, it has been shown that implementing a swarm intelligence algorithm of image edge detection with circuits of memristors consumes less energy than conventional methods [18]. Related evolution of graphs implemented by memristors has been simulated in Refs. [35–37] for closed systems with internal voltage supplies. The analogy between circuits of memristors and the unicellular organism *Physarum*, as well as its application to computing, has been studied [38–40]. Exploring other swarm intelligence ideas and different substrates to implement them is an exciting road toward low-energy-cost systems that perform complex optimization tasks.

ACKNOWLEDGMENTS

The authors acknowledge A. Madhavan, M. Daniels, N. Zhitenev, J. McClelland, R. McMichael, S. Dushenko, and P. Shrestha for helpful comments and discussions. A.M. acknowledges support under the Cooperative Research Agreement between the University of Maryland and the National Institute of Standards and Technology, Center for Nanoscale Science and Technology, Grant No. 70NANB10H193, through the University of Maryland. This material is based upon work supported by the National Institute of Standards and Technology Summer Undergraduate Research Fellowship (SURF) Program under Grant No. 70NANB18H080.

[1] E. W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. **1**, 269 (1959).

[2] R. K. Ahuja, K. Mehlhorn, J. Orlin, and R. E. Tarjan, Faster algorithms for the shortest path problem, J. ACM **37**, 213 (1990).

[3] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, Shortest paths algorithms: Theory and experimental evaluation, Math. Program. **73**, 129 (1996).

[4] M. Thorup, Undirected single-source shortest paths with positive integer weights in linear time, J. ACM **46**, 362 (1999).

[5] M. Thorup, Integer priority queues with decrease key in constant time and the single source shortest paths problem, J. Comput. Syst. Sci. **69**, 330 (2004).

[6] M. L. Fredman and R. E. Tarjan, in *25th Annual Symposium on Foundations of Computer Science, 1984* (IEEE, Piscataway, 1984), p. 338.

[7] R. Williams, in *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, STOC '14* (ACM, New York, NY, USA, 2014), p. 664.

[8] E. Bonabeau, M. Dorigo, and G. Theraulaz, Inspiration for optimization from social insect behaviour, Nature **406**, 39 (2000).

[9] M. Dorigo, M. Birattari, and T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. **1**, 28 (2006).

[10] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, Data mining with an ant colony optimization algorithm, IEEE Trans. Evol. Comput. **6**, 321 (2002).

[11] C. Blum, Ant colony optimization: Introduction and recent trends, Phys. Life Rev. **2**, 353 (2005).

[12] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, A comprehensive review of swarm optimization algorithms, PLoS ONE **10**, e0122827 (2015).

[13] F. Ducatelle, G. A. Di Caro, and L. M. Gambardella, Principles and applications of swarm intelligence for adaptive routing in telecommunications networks, Swarm Intell. **4**, 173 (2010).

[14] R. Poli, J. Kennedy, and T. Blackwell, Particle swarm optimization, Swarm Intell. **1**, 33 (2007).

[15] Y. V. Pershin and M. Di Ventra, Solving mazes with memristors: A massively parallel approach, Phys. Rev. E **84**, 046703 (2011).

[16] Y. V. Pershin and M. Di Ventra, Self-organization and solution of shortest-path optimization problems with memristive networks, Phys. Rev. E **88**, 013305 (2013).

[17] Y. V. Pershin and M. Di Ventra, Memcomputing implementation of ant colony optimization, Neural Process. Lett. **44**, 265 (2016).

[18] Z. Pajouhi and K. Roy, Image edge detection based on swarm intelligence using memristive networks, IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst. **37**, 1774 (2018).

[19] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, The missing memristor found, Nature **453**, 80 (2008).

[20] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, Switching dynamics in titanium dioxide memristive devices, J. Appl. Phys. **106**, 074508 (2009).

[21] S. Menzel, M. Waters, A. Marchewka, U. Böttger, R. Dittmann, and R. Waser, Origin of the ultra-nonlinear switching kinetics in oxide-based resistive switches, Adv. Funct. Mater. **21**, 4487 (2011).

[22] T. Chang, S.-H. Jo, K.-H. Kim, P. Sheridan, S. Gaba, and W. Lu, Synaptic behaviors and modeling of a metal oxide memristive device, Appl. Phys. A **102**, 857 (2011).

[23] M. H. R. Lankhorst, B. W. S. M. M. Ketelaars, and R. A. M. Wolters, Low-cost and nanoscale non-volatile memory concept for future silicon chips, Nat. Mater. **4**, 347 (2005).

[24] D. Ielmini and A. L. Lacaita, Phase change materials in non-volatile storage, Mater. Today **14**, 600 (2011).

[25] S. Lai, in *IEEE International Electron Devices Meeting 2003* (IEEE, Piscataway, 2003), p. 10.1.1.

[26] R. Bez, in *2009 IEEE International Electron Devices Meeting (IEDM)* (IEEE, Piscataway, 2009), p. 1.

[27] D. J. Watts and S. H. Strogatz, Collective dynamics of 'small-world' networks, Nature **393**, 440 (1998).

[28] R. Albert, H. Jeong, and A.-L. Barabási, Internet: Diameter of the world-wide web, Nature **401**, 130 (1999).

[29] D. S. Bassett and E. Bullmore, Small-world brain networks, Neuroscientist **12**, 512 (2006).

[30] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature **435**, 814 (2005).

[31] D. B. Strukov and K. K. Likharev, CMOL FPGA: A reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices, Nanotechnology **16**, 888 (2005).

[32] D. Strukov and A. Mishchenko, in *Proceedings of the Conference on Design, Automation and Test in Europe, DATE '10* (European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 2010), p. 661.

[33] A. Madhavan, T. Sherwood, and D. B. Strukov, High-throughput pattern matching with CMOL FPGA circuits: Case for logic-in-memory computing, IEEE Trans. Very Large Scale Integration (VLSI) Syst. **26**, 2759 (2018).

[34] R. M. Wang, C. S. Thakur, and A. van Schaik, An FPGA-based massively parallel neuromorphic cortex simulator, Front. Neurosci. **12**, 213 (2018).

[35] F. Caravelli, F. L. Traversa, and M. Di Ventra, Complex dynamics of memristive circuits: Analytical results and universal slow relaxation, Phys. Rev. E **95**, 022140 (2017).

[36] F. Caravelli, Asymptotic behavior of memristive circuits and combinatorial optimization, arXiv:1712.07046 [cond-mat,physics:physics] (2017).
[37] F. Caravelli, Locality of interactions for planar memristive circuits, Phys. Rev. E **96**, 052206 (2017).
[38] Y. V. Pershin, S. La Fontaine, and M. Di Ventra, Memristive model of amoeba learning, Phys. Rev. E **80**, 021926 (2009).
[39] V. Ntinas, I. Vourkas, G. Ch Sirakoulis, and A. I. Adamatzky, Modeling Physarum space exploration using memristors, J. Phys. D: Appl. Phys. **50**, 174004 (2017).
[40] E. Gale, A. Adamatzky, and B. de Lacy Costello, Slime mould memristors, BioNanoScience **5**, 1 (2015).