Shortest Path Computing in Directed Graphs with Weighted Edges Mapped on Random Networks of Memristors

Carlos Fernandez[1], Ioannis Vourkas[1], and Antonio Rubio[2]

[1]*Department of Electronic Engineering, Universidad Tecnica Federico Santa Maria, Valparaíso, P.C. 2390123, CHILE*

[2] *Dept. of Electronic Engineering, Universitat Politècnica de Catalunya, Barcelona, P.C. 08034, SPAIN*

Abstract

To accelerate the execution of advanced computing tasks, in-memory computing with resistive memory provides a promising solution. In this context, networks of memristors could be used as parallel computing medium for the solution of complex optimization problems. Lately, the solution of the shortest-path problem (SPP) in a two-dimensional plane has been given wide consideration. Some still open problems in such computing approach concern the time required for the network to reach to a steady state, and the time required to read the final result, stored in the state of a subset of memristors that represent the solution. This paper presents a circuit simulation-based performance assessment of memristor networks as SPP solvers. A previous methodology was extended to support weighted directed graphs. We tried memristor device models with fundamentally different switching behavior to check their suitability for such applications and the impact on the timely detection of the solution. Furthermore, the requirement of binary vs. analog operation of memristors was evaluated. Finally, the memristor network-based computing approach was compared to known algorithmic solutions to the SPP over a large set of random graphs of different sizes and topologies. Our results contribute to the proper development of bio-inspired memristor network-based SPP solvers.

Keywords

Memristor network, resistive switching, shortest path, computational memory, resistive computing.

1. Introduction

The emerging technology of resistive switching devices (memristors) [1] has attracted considerable attention from academia and industry owing to its potential impact in different domains of electronics [2] - [4]. In this context, the overall response of memristors, when they are organized into networks, gained an ever-increasing interest after Pershin and Di Ventra demonstrated that this could lead to a new analog computing approach which they named “memcomputing” [5], [6]. Several potential applications proposed for memristor networks include memory, logic, and signal processing tasks [7] – [9]. They were particularly found promising candidates as computing medium for the solution of complex optimization problems in a massively parallel fashion [10]. Moreover, the result of the computation is recorded into the resistive states of the memristors, which infers *in-memory* computations [11].

Building upon Pershin and Di Ventra’s initial idea, inspired by biological processes [12], the maze-solving and more generally the problem of computing the shortest-path (SPP) in a two-dimensional plane, using memristor networks, have been given wide consideration lately [13] – [16]. Following

this approach, any graph is represented by a network of memristors where the vertices (nodes) of the graph are electrical junctions interconnected by memristive configurations. The shortest path computation emerges from the polarity-dependent dynamical evolution of the resistive states of the memristors without any supervision. However, some still open problems in such computing approach concern: (i) *the time required for the network to reach to a steady state*. This depends on the properties of the applied signal, the network topology, and the *switching properties of the memristors*. There are memristors that switch rapidly in a binary fashion, while others change their state in an analog manner. (ii) *The time required to read the result*, stored in the state of a subset of memristors that represent the solution. A case study of the performance of memristor networks as SPP solvers w.r.t. the memristor device properties and the graph topological features, is still missing.

This paper extends the preliminary results presented in [17] to fill this gap with a circuit simulation (LTSPICE)-based performance assessment of memristor networks as SPP solvers. A previous methodology [15] was here extended to support mapping of weighted directed graphs onto networks of memristors. Moreover, we tested the validity of a previously proposed criterion in [18] for the detection of a solution to an SPP and also tried memristor models with fundamentally different switching behavior, to check their suitability for such massively parallel computing applications that involve the interconnection of a large number of memristors. Furthermore, the preliminary evaluation of the impact of fast resistive switching response on the final outcome of the computation led us to explore further the requirement for *binary* vs. *analog* operation of memristors. Eventually, the performance of the memristor-network based SPP solver was compared to known algorithmic solutions to the SPP executed over a set of large random graphs. The presented simulation results confirm the advantages of such a circuit-based approach and thus contribute to the proper development of bio-inspired memristor network-based parallel SPP solvers.

2. Computing Platform Description

- 2.1 Memristor Grid & Simulation Setup

The operation of such computing platform consists in three stages: *initialization*, *computation*, and *reading* of the result. The memristor device configurations representing the graph edges usually consist in two bipolar memristors with opposite polarity connected in parallel (commonly described as anti-parallel connection), so that the graph edges are (i) independent of the polarity of the flowing current and (ii) electrically equivalent (all edges are undirected and have the same weight, i.e. the same composite resistance, which is the equivalent resistance of the parallel memristors) [15].

The computing platform is initialized with all memristors in the high resistive state (HRS, OFF state, or R_{OFF}). Finding the shortest path between any two nodes of the circuit requires connecting them to a voltage source and ground, respectively. More current flows through the shortest branch in the network which is the more conductive path; i.e. it is composed of the lowest number of connected memristive edges. Depending on the polarity of the voltage drop at their terminals, in each anti-parallel connection of memristors only the device which is forward-biased will switch to a low resistive state (LRS, ON state, or R_{ON}) (see Fig. 1h). As a consequence, the composite resistance will change from a high ($R_{OFF}||R_{OFF}$) to a low ($R_{OFF}||R_{ON}$) value, where operator || means parallel

connection of resistances. The system explores all paths simultaneously in parallel and the memristors in the shortest branch are the first to complete the change of state towards R_{ON}, thus marking the subset of edges that altogether compose the solution to the problem, if any. This difference in the composite resistance in the memristive edges that form the SPP solution makes possible reading out the result of computation. However, the devices forming the final SPP solution will not necessarily switch their state simultaneously. Generally, the time required for the network to reach to a solution depends on the *applied voltage*, the *network topology*, and the *properties of the employed memristors*.

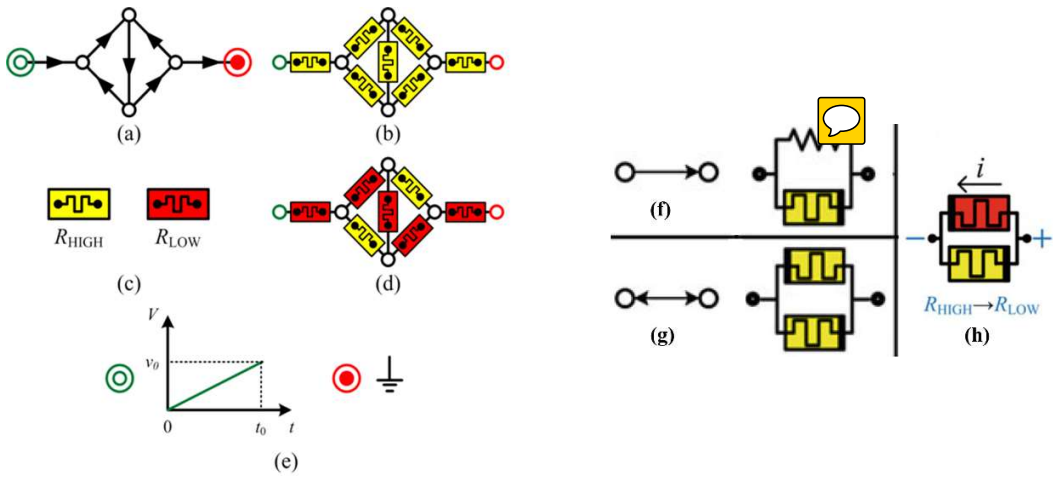
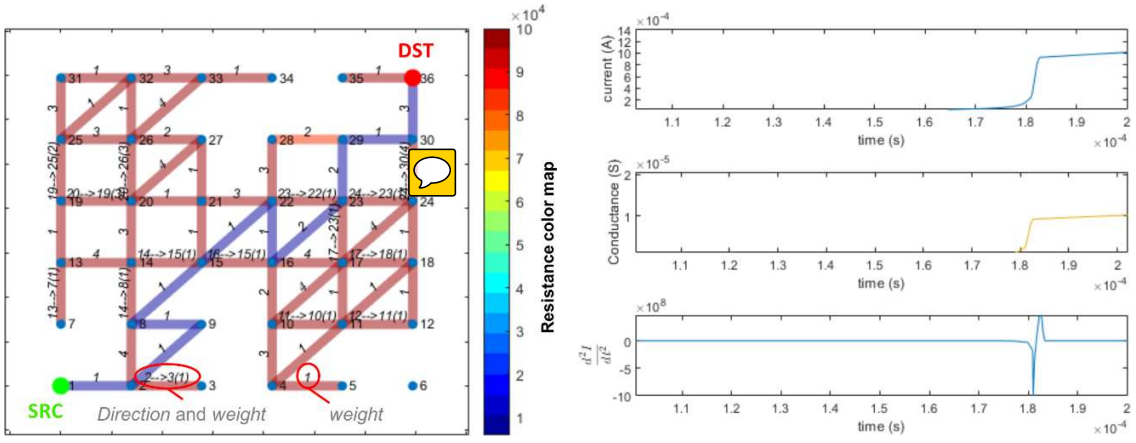


Fig. 1 (a, b) Example of a directed graph mapped to a memristive network. (c, d) The solution to the SPP marked with memristive components shown in red color. (e) The input voltage applied to the SRC and the ground applied to the DST node. (f, g) Mapping correspondence for directed and undirected graph edges according to [15]. (h) Polarity-independent switching of antiparallel memristors emulating an undirected edge of a graph.

Fig. 1(a,b) show a directed graph example mapped onto a memristor network, whereas Fig. 1(c-e) show the operational details of the computing medium. Fig. 1(f-h) show the mapping of graph edges to the electrically equivalent memristive connections and the way that anti-parallel memristors enable the desired polarity-independent switching in undirected graph edges. When only directed edges are present in a graph, then a single memristor is enough to represent every edge. On the contrary, when both directed and undirected edges are present in a graph, then we use a resistor of value equal to R_{OFF} in parallel with a memristor to emulate the directed edges, while anti-parallel memristors are used for the undirected edges. This way, since all connections are of equal (unitary) weight in the graph, they are initially electrically equivalent with composite resistance equal to $R_{OFF}||R_{OFF}$ (being the result of either two parallel memristors or a memristor and a resistor).

In the context of this work, a Matlab© script was prepared for the configurable creation of different graph examples based on a fixed square grid topology with randomly removed edges, as shown in Fig. 2(a). The script accepts information such as the source/destination nodes, the applied voltage, the memristor model to be used and the values for its parameters, the total simulation time, and also some properties of the target graph to be created, concerning the edge weights and their orientation. The memristor network corresponding to the graph is finally described in a netlist that we simulate using the LTSPICE© circuit simulation tool. Once the simulation is over, the results are processed in Matlab© to visualize the final network state and information relevant to the evolution of the computation, such as the total conductance of the network (see in Fig. 2(b)).

Fig. 2(a) shows a visualization of the tested graph and, at the same time, of the network state in a simulation scenario for the shortest path computation between SRC (bottom left) and DST (top right) nodes in the grid. For such simulation, we used the threshold-type switching model of a bipolar voltage-controlled memristor proposed in [19] (see description in section 3). Likewise we showed in Fig. 1(e), a ramped voltage is applied to the SRC node as the length of the shortest path is not known in advance, thus the minimum required voltage amplitude is unknown either. The composite resistance of every link is shown using a linear color scale and the network evolution follows the directed connectivity of the graph. The numbers next to every link indicate the *weight* and *direction*; the latter is shown as "$x \rightarrow y$" for nodes x and y only for directed edges, along with the integer weight indicator shown in parenthesis. In spite of the complexity of the route, the network gradually converges to the SPP solution and the low-resistance path is visually observed, contrasting with the rest of the edges which are still in a much higher composite resistance. By observing Fig. 2(a), it can be noticed that the direction of edges is correctly considered (see e.g. 23→22) as well as the edge weight, since memristors found in alternative connections with a higher total weight remain in HRS, as expected.



(a) (b)

Fig. 2 (a) Visualization of the simulation results for a SPP computation from node 1 (SRC) to 36 (DST) in a random graph mapped on a 6×6 grid, while applying a voltage ramp from 0 to 100V in 1ms, using the memristor model of [19] with [R_{ON}, R_{OFF}] = [2, 200]KΩ (see description in section 3). (b) Evolution of the input current through the network with time, the global conductance of the network, and the second time derivative of the input current. The network state in (a) corresponds to the exact moment when the SPP solution was detected.

- 2.2 Preliminary Performance Screening & Shortest Path Detection

A higher voltage ramp speed (VRS) of the applied voltage ramp leads to faster computations since the minimum required voltage to produce a result is reached faster. Note that only a certain range of input voltage will lead to a successful computation of the shortest path. According to [18], *the optimal input voltage is proportional to the length of the shortest path*, thus being independent of the size of the graph. The increase in the conductance of the memristive edges belonging to the shortest path causes an increase in the global conductance of the network. A useful means to detect a solution to the SPP corresponds to observing a sudden drop in the second time derivative of the current [18]. This can be seen in the bottom graph shown in Fig. 2(b). Once this drop is detected, the input is removed and the result is read out. Starting from the SRC node, the conductance of each connecting

edge is measured. The next node on the shortest path is the one connected to the current node through the most conductive memristive edge. This process is repeated until the DST node is reached. Therefore, reading the solution implies probing only a fraction of the total edges in the memristive network. This notwithstanding, reading correctly the solution requires the measured edge belonging to the shortest path to be *notably more conductive* than the rest of the adjacent edges, otherwise the reading process might erroneously follow degenerate/partially formed paths.

It is worth mentioning that this detection method based on observing the second time derivative of the current, works well only for the very first discovered path(s). Note that in memristor network-based SPP computations [13]-[15], while the shortest solution(s) is(are) the first to emerge, other existing solution paths which are longer than the first, might be revealed later while the input voltage is still increasing. Fig. 3 shows such a case where an SPP solution was found in Fig. 3(a) once the drop in the second time derivative of the current was detected. However, later in the same simulation another solution was found as shown in Fig. 3(b), which concerns a longer path than the first one. We notice that the change in the current derivative curve is not as intense as in the first case. This makes more difficult the choice of a proper threshold value for the current derivative such that, any drop below that value will indicate correctly the detection of a new solution.

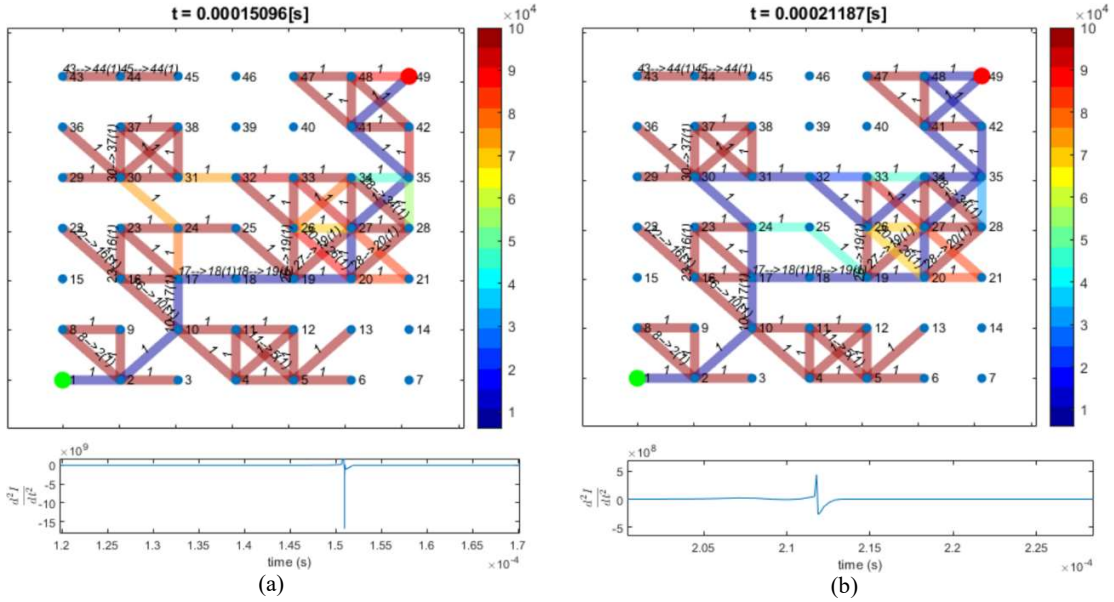


Fig. 3 Visualization of the simulation results for an SPP computation at two different moments with (b) being a posterior phase of the same simulation shown in (a). The problem concerns a path from node 1 to 49 in a random directed graph on a 7×7 grid, while applying a voltage ramp from 0 to 100V in 1ms. Memristors were simulated using the memristor model of [19] with [R_{ON}, R_{OFF}] = [2, 200]KΩ (see description in section 3). The time when a solution was detected is shown at the top. At the bottom, the evolution of the 2nd time derivative of the current is shown.

This is equally important not only to void missing existing solutions but also to avoid the erroneous detection of incomplete paths that are not solutions to the SPP but might cause misread fluctuations in the current derivative curve. We show such an example in Fig. 4 where we simulated a directed graph without any connection between the SRC and DST nodes. A low-resistive path is partially formed connecting SRC with node 33. However, because of the direction of the three edges connecting nodes 33 and DST, namely 34→33, 35→34 and DST→35, the respective memristors in

these connections correctly remain in HRS. Nevertheless, in the current derivative curve we observe large fluctuations occurring during the computation which could be potentially misunderstood indicating the detection of an SPP solution while there isn´t any path connecting SRC and DST. Everything considered, our simulation results on random graphs highlight that a proper selection of a threshold value for the second time derivative of the current is necessary to make more effective such an, otherwise useful, detection criterion in memristor network-based SPP computations.

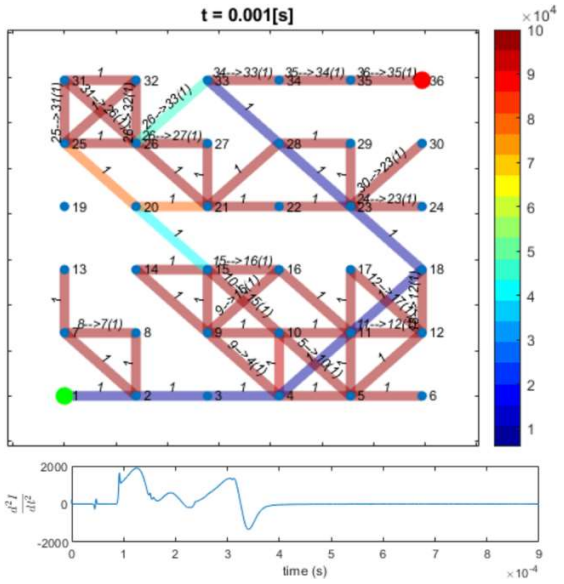


Fig. 4 Visualization of the simulation results after 0.001s for an SPP computation concerning the search of a path from node 1 to 36 in a random directed graph defined on a 6×6 grid, while applying a voltage ramp rising from 0 to 100V in 1ms. Memristors were simulated using the memristor model of [19] with $[R_{ON}, R_{OFF}] = [2, 200]$KΩ (see description in section 3). At the bottom, the evolution of the 2nd time derivative of the current is shown.

3. Memristor Device Models under Consideration

The development of compact device models contributed significantly to the progress of research in memristor technology [20], [21], being adequate to capture the experimentally observed resistive switching behavior and to demonstrate functionality/applicability of memristors in potential applications. One of the objectives of this study concerns revealing the desired properties of the memristor devices to be used in such massively parallel computing circuits. This, in turn, at simulation level corresponds to finding the desired device properties while probing with different device models. In previous sections, all the simulation results were based on the behavioral threshold-type model of a voltage-controlled bipolar memristor proposed by Pershin *et al.* in [19]. We selected such a threshold-based model because the existence of switching thresholds is a very important common feature of most memristor device types. However, in this section we describe in more details four selected models with fundamentally different switching behavior, widely used in the recent literature, which we later explore further to evaluate their impact on the overall performance of the memristor network in SPP computations.

The first model to describe is the behavioral threshold-type model used so far in this work [19], which has a voltage-dependent switching rate, and is described by the following equations:

$$i_m(t) = x^{-1} \cdot v_m(t) \qquad (1)$$

$$\dot{x} = f(v_m) \cdot \left[\theta(v_m) \cdot \theta(R_{OFF} - x) + \theta(-v_m) \cdot \theta(x - R_{ON})\right] \qquad (2)$$

$$f(v_m) = \beta \cdot v_m + 0.5 \cdot (\alpha - \beta) \cdot \left[\,|v_m + v_t| - |v_m - v_t|\,\right] \qquad (3)$$

$v_\mathrm{m}(t)$ and $i_m(t)$ denote the voltage and current across the memristor, respectively, whereas $x \equiv R_\mathrm{m}$ is the state variable which corresponds to the resistance/memristance of the device. v_t defines the SET ($R_\mathrm{m} \rightarrow R_\mathrm{ON}$) and RESET ($R_\mathrm{m} \rightarrow R_\mathrm{OFF}$) threshold voltages, which here are assumed both equal to $|v_t|$ without loss of generality. R_ON and R_OFF are the limiting values of R_m. The θ-functions in (2) are step functions which limit R_m between R_ON and R_OFF. Parameters α and β ($\alpha << \beta$) define the rate of change or R_m at the regions where $|v_\mathrm{m}| < v_t$ and $|v_\mathrm{m}| > v_t$, respectively; i.e. these coefficients define the slope of the curve given by (3) below and above the threshold $|v_t|$ for SET/RESET operations. By slightly modifying (3) we can define asymmetric thresholds $v_{t,\mathrm{SET}} \neq v_{t,\mathrm{RESET}}$ and different switching rates for the SET and RESET ($\beta_\mathrm{SET} \neq \beta_\mathrm{RESET}$) processes. According to (3), the switching rate depends linearly only on the voltage drop v_m and it remains constant for a constant v_m. This model also allows defining directly the R_ON and R_OFF values by updating the corresponding parameters.

Another model of interest for this work is the linear model, first proposed by Hewlett-Packard in 2008 [22] and later developed in SPICE by Biolek *et al.* [23], which has a current-dependent switching rate. This model has a normalized state variable x whose value varies in the range [0, 1] and it is described by the following equations:

$$v_m(t) = R_m(x) \cdot i_m(t) \qquad (4)$$

$$R_m(x) = R_{ON} \cdot x + R_{OFF} \cdot (1 - x) \qquad (5)$$

$$\dot{x} = \left(\frac{R_{ON} \cdot \mu_v}{D^2}\right) \cdot i_m \cdot f(x, i_m) = k \cdot i_m \cdot f(x, i_m) \qquad (6)$$

We included the *state-dependent Ohm´s Law* shown in (1) again in (4), just to highlight that this is a *current-controlled* model, whereas (1) referred to a *voltage-controlled* model. Moreover, we observe that here the memristance R_m is a function of the state variable as given in (5) and the switching rate, given by (6), is proportional to the current flowing through the memristor i_m and depends on some device-specific material/geometrical properties such as the dopant mobility μ_v and the width of the metal-oxide structure D. In (6) we also incorporated a window function $f(\cdot)$, which is a multiplicative

factor whose role is to slow down the change of a state variable as it approaches a boundary value, in fact aiming to model nonlinear dopant drift phenomena (read further in [24] about the role of window functions in memristor modeling). This model also allows defining directly the R_{ON} and R_{OFF} values by updating the corresponding parameters. In previous relevant works on memristor network-based SPP computations [13], the simulations were based on current-controlled memristors and the authors observed a self-reinforcement of the shortest path solution with time. In fact, the least resistive path distinguished faster by attracting more and more current. However, this reinforcement is true only if the memristance change rate is proportional to the current through the device, as is the case with the linear HP model, but it does not apply to voltage-controlled memristors. So, this is the main reason why this model was included in this study.

Furthermore, we examined another threshold-type model with a voltage-dependent switching rate, which was proposed by *Knowm Inc*. to model behavior of their commercial devices [25]. It is a semi-empirical model known as the "mean metastable switch" memristor model and it describes a memristor as a collection of metastable switches (MSS) where each switch can be in either one or the other state. By MSS we refer to an idealized two-state element that switches probabilistically between its two states as a function of applied voltage and temperature (for all the model equations see further in [26]). The model uses a *state-dependent Ohm's Law* like in (1) and computes the device conductance using an equation similar to (5) but in terms of conductances G_{ON} and G_{OFF}, via a dimensionless state variable x whose value varies in [0, 1]. Most importantly, the switching-rate of the state variable x is given by the following equation:

$$\dot{x} = \frac{1}{\tau} \cdot \left(\frac{1}{1 + e^{-\beta \cdot (v_m - V_{ON})}} \cdot (1 - x) - \left(1 - \left(\frac{1}{1 + e^{-\beta \cdot (v_m + V_{OFF})}} \right) \right) \cdot x \right) \quad (7)$$

where τ is a time constant and $\beta = q/kT$. Likewise in (2) and (3), we observe again in (7) the role of the voltage thresholds in the overall switching behavior, defined here as V_{ON}/V_{OFF} for SET/RESET, respectively. However, the dependency of dx/dt both on the state variable and on the applied voltage in a nonlinear manner, makes this an interesting alternative approach to explore, compared to the much simpler model described by (1)-(3). This model also allows defining directly the R_{ON} and R_{OFF} values by updating the corresponding parameters.

The last model included in this comparison was the model proposed by Yakopcic *et al*. in [27], which belongs to the category of *hyperbolic sine models* and was developed based on a more general understanding of memristor dynamics. According to [27], the hyperbolic sinusoid function can be used to approximate quite well the *i-v* relationship of metal-insulator-metal (MIM) memristive structures. Therefore, it was found reasonable to explore this kind of expressions in several modeling works as detailed in [20], given that memristor devices are commonly fabricated in a MIM structure. The generalized *i-v* relationship for this memristor model is shown in (8):

$$i_m(t) = \begin{cases} a_1 \cdot x \cdot \sinh(b \cdot v_m(t)) & , \quad v_m(t) \geq 0 \\ a_2 \cdot x \cdot \sinh(b \cdot v_m(t)) & , \quad v_m(t) < 0 \end{cases} \quad (8)$$

The state variable x takes values between 0 and 1 and directly impacts the conductivity of the device. However, defining directly the R_{ON} and R_{OFF} limiting values is not possible since there are not such parameters in the model. There is a unique parameter b used independent of the voltage polarity, but two different amplitude parameters α_1 and α_2 that make possible tuning conductivity differently during SET and RESET. The change in the state variable x is based on two different functions, namely, $g(\cdot)$ and $f(\cdot)$, as shown in (9), where the $\eta = \pm 1$ is a parameter used to determine the direction of the dynamic state variable motion relative to the voltage polarity.

$$\dot{x} = \eta \cdot g(v_m) \cdot f(x) \quad (9)$$

The function $g(\cdot)$, that is given by (10), imposes SET/RESET voltage thresholds on the memristor model with the possibility of having asymmetric values: V_{ON} for SET with positive voltages and V_{OFF} for RESET with negative voltages. The parameters A_{ON} and A_{OFF} determine how quickly the state of the device changes once the aforementioned threshold voltages have been exceeded.

$$g(v_m) = \begin{cases} A_{ON} \cdot \left(e^{v_m} - e^{V_{ON}}\right) & , \quad v_m > V_{ON} \\ -A_{OFF} \cdot \left(e^{-v_m} - e^{V_{OFF}}\right) & , \quad v_m < -V_{OFF} \\ 0 & , \quad otherwise \end{cases} \quad (10)$$

The other function used to model the state variable dynamics is $f(\cdot)$ and it is a window function that was included based on similar assumptions as mentioned previously in (6) (for all the model equations see further in [27]). All in all, this model is a threshold-type switching model of a voltage-controlled memristor but has an increased complexity and richer dynamics compared to the rest of threshold-type models mentioned previously, and it has been successfully applied to match characterization data from several different materials and device structures [20]. For all these reasons, it was included in this study.

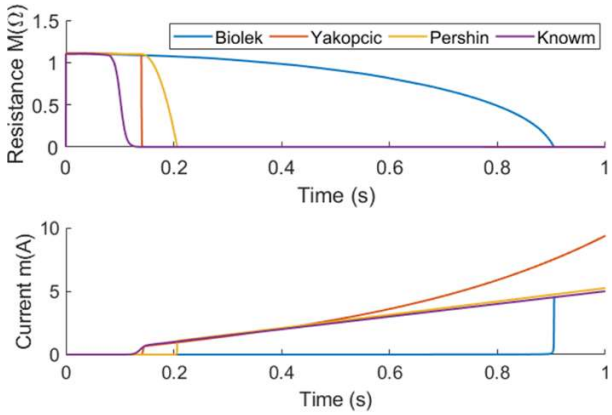


Fig. 5 Simulation of the SET process of a memristor from 1.1MΩ to 1KΩ, under a ramped applied voltage which rises from 0V to 5V in 1s. Simulation used the model of Pershin [19] with parameters $\alpha = 0$, $\beta = 0.0001e12$, $vt = 0.7V$; the linear model by HP (Biolek) [23] with parameters D = 12nm, μ_v = 40e-15 $m^2s^{-1}V^{-1}$, p=5; the *Knowm Inc*. model [25] with parameters τ=0.003, T=298.5, V_{ON}= V_{OFF} = 0.7V; and the model of Yakopcic [27] with parameters $a_1 = 0.002$, $a_2 = 0.002$, $b = 0.45$, V_{ON} = V_{OFF} = 0.7V, and A_{ON} = A_{OFF} = 10000.

For a fairer comparison, the values of the parameters of the models were selected in such a way so that all of them had the same SET/RESET voltage thresholds ±0.7V (when applicable), showed a similar switching time under the same applied voltage, and exhibited the same resistance window between 1KΩ and 1,1MΩ. In our case, such a resistance range was commonly selected for all models considering the default range exhibited in the model by Yakopcic *et al*. [27] which is the most difficult to adjust, since it has not R_{ON} and R_{OFF} parameters. Fig. 5 shows R_m-t and i_m-t curves for all the considered models during the application of a voltage ramp that rises from 0 to 5V in 1s. We observe that the threshold-type models share similar characteristics in their behavior, with a small difference in their switching time. However, the linear current-controlled memristor model (Biolek) takes longer to complete the SET switching, which is reinforced by the flowing current; i.e. the larger the current, the faster the switching rate. Making this model faster requires changing parameter k in (6) which depends on device-specific material/geometrical properties which we rather kept at their default values suggested in [23]. On the other hand, slowing down the switching process of the other three models is easier but it could impact the performance of the network as we comment in the following sections. Moreover, we observe the nonlinear LRS state of the Yakopcic model owing to (8), compared to the Ohmic LRS of the rest of the models which we mentioned previously in this section.

4. Impact of Device Properties in Memristor Network-based SPP Computations

In all simulation scenarios, the edges of the randomly generated graphs have integer weights, which we represent in the circuit by connecting in series multiple instances of the same memristive configuration shown in Fig. 1(f, g). Thus, an *undirected* edge with *weight* = 3 corresponds to 3 pairs of anti-parallel memristors, connected in series. This way, the minimum required voltage to force the forward polarized memristors of that branch to switch to R_{ON} is 3× higher than the voltage required for a branch of *weight* = 1, owing to the voltage divider effect while also assuming identical memristors in both branches. Likewise, a *directed* edge with *weight* = 2 corresponds to two pairs of parallel memristor-resistor connected in series, where the polarity of memristors is in line with the edge direction, as shown in Fig. 1(f). According to [15], weighted edges could be emulated by combining memristors with different SET/RESET voltage thresholds. However, such approach is not universal as it would apply only to threshold-based models/devices. On the contrary, the proposed here approach to represent weighted edges is simple and straightforward to implement, as well as universal, and it was proved effective in several simulation scenarios with all the considered models.

Unlike mentioned in [18], in this work we demonstrate that not all types of switching behavior are equally suitable for such network-based massively parallel computations. Our simulation-based study highlighted that the improper selection of the target memristor devices (as reflected by the respective properties of the device models) could have a negative effect on the *computation time*, on the *detection of a solution*, as well as on the final *read-out process* of the devices on the shortest path. We simulated several cases and Fig. 6 summarizes the general tendencies, highlighting the influence of the used model in the detection of the solution using the approach based on the second time derivative of the current. It should be noted that a solution was always found regardless of the model we used. Moreover, memristor variability was not considered in our simulations, but it is generally

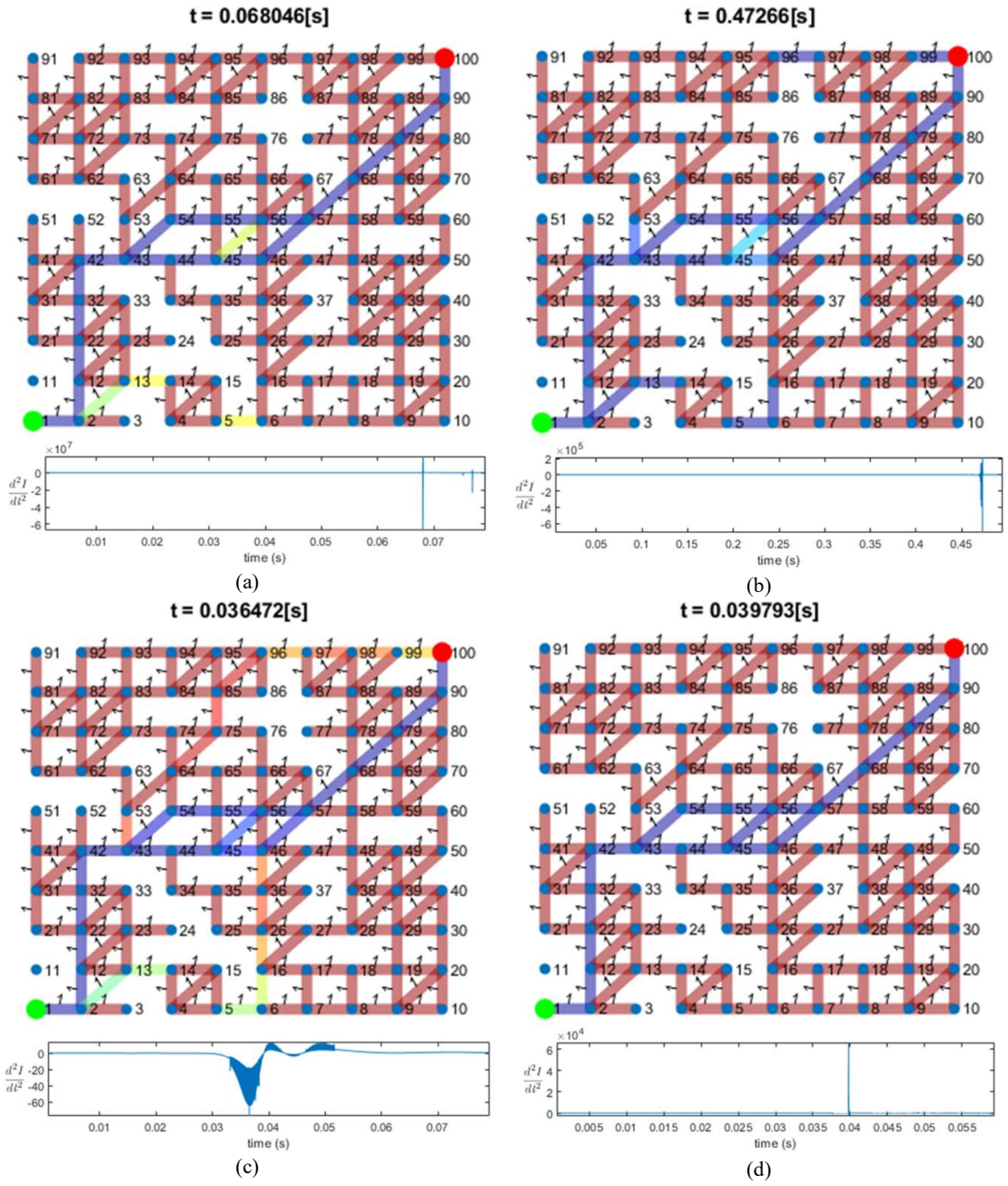


Fig. 6 Visualization of the simulation results for an SPP computation from node 1 to 100, in a random undirected graph on a 10×10 grid under the application of a voltage ramp that rises from 0 to 130V in 0.5s, using (a) the model of Pershin [19], (b) the linear HP model [23], (c) the model of *Knowm Inc*. [25], and (d) the model of Yakopcic [27], with the same parameters mentioned for Fig. 5. The time when a solution was detected is shown at the top. At the bottom, the evolution of the 2nd time derivative of the current is shown. The color scale for the memristance is the same as in previous similar figures.

considered an asset in this context, as it inherently helps the network to choose one shortest path when several equivalent solutions are present [15], [18]. Note that all models led to the formation of the unique solution, which in this case includes two equivalent alternative paths connecting node 43 with 57.

Comparing the performance of voltage-controlled threshold type models with the linear current-controlled model, we notice in the latter that some partially formed low-resistive paths emerged, as shown in Fig. 6(b), which generally complicate the read-out phase. For voltage-controlled models, we noticed there is a trade-off between the applied VRS and the switching speed of memristors. Generally, if the memristors switch too fast their state, or if they are sufficiently slow, then partially formed low-resistive paths could emerge. In either case, modifying the applied VRS can compensate towards a better performance; e.g. apply a higher/lower VRS for slower/faster memristors. Another important issue concerns the timely detection of the solutions found during computations, where the form of the current time derivative curve could affect significantly. In the case of the *Knowm Inc*. model, we notice in Fig. 6(c) that fluctuations in the current time derivative reach only small values, compared to what can be observed for the rest of the models. Therefore, likewise we mentioned in section 2.2, a proper selection of a threshold value for the second time derivative of the current is necessary to make more effective the detection criterion. As far as computation time is concerned, the two models where the switching speed depends exponentially on the applied voltage result the fastest ones, as expected. Similar tendencies were observed in several more simulated graphs (not shown here to avoid redundancy), which led us to the conclusion that (i) faster memristors are more beneficial in terms of *computation time* and *precision*, and (ii) threshold-type switching is better than linear switching.

Given that the threshold-type voltage-controlled models which demonstrated superior performance so far, switch their state sufficiently fast to assume they behave in a binary manner, we studied further the requirement of the analog nature of memristors for this computing application. Our objective was to figure out whether filamentary memristors that switch in a binary fashion between the HRS and LRS states could be more appropriate. To this end, we modified the model of Pershin [19] so that, once the voltage drop on the device terminals exceeds the threshold voltage, it switches instantly to the opposite boundary resistive state. We then compared the performance of the network using binary switching devices vs. the original threshold type memristor model which used so far.

As it can be observed in the simulation results shown in Fig. 7, concerning a weighted directed graph with graph weights varying between 1 and 4, even though fast switching is generally desired, the analog nature of memristors is still important for such computing tasks. As expected, binary switching leads to much faster computations. Moreover, when binary switching devices are concerned, the detection of a solution using the second time derivative of the current, works fine. However, we observed in several cases that the final network state always had many edges that were not part of the SPP solution but still had changed to low resistance. For instance, Fig. 7(a, c) show the network performance when the original memristor model was used, where we observe again a unique correct solution to the SPP, respecting both edge directions and their weights. On the other hand, Fig. 7(b, d) correspond to the binary resistive switch model, where we observe the much faster response at the top of the figure. However, several nonshortest paths appeared. Branches with small differences in the total weight switch ON altogether simultaneously because the instant voltage drop they momentarily receive is enough to immediately change all the involved devices, thus creating erroneous solutions to the SPP or, sometimes, simply giving a bunch of partially formed paths that complicate the read-out phase.

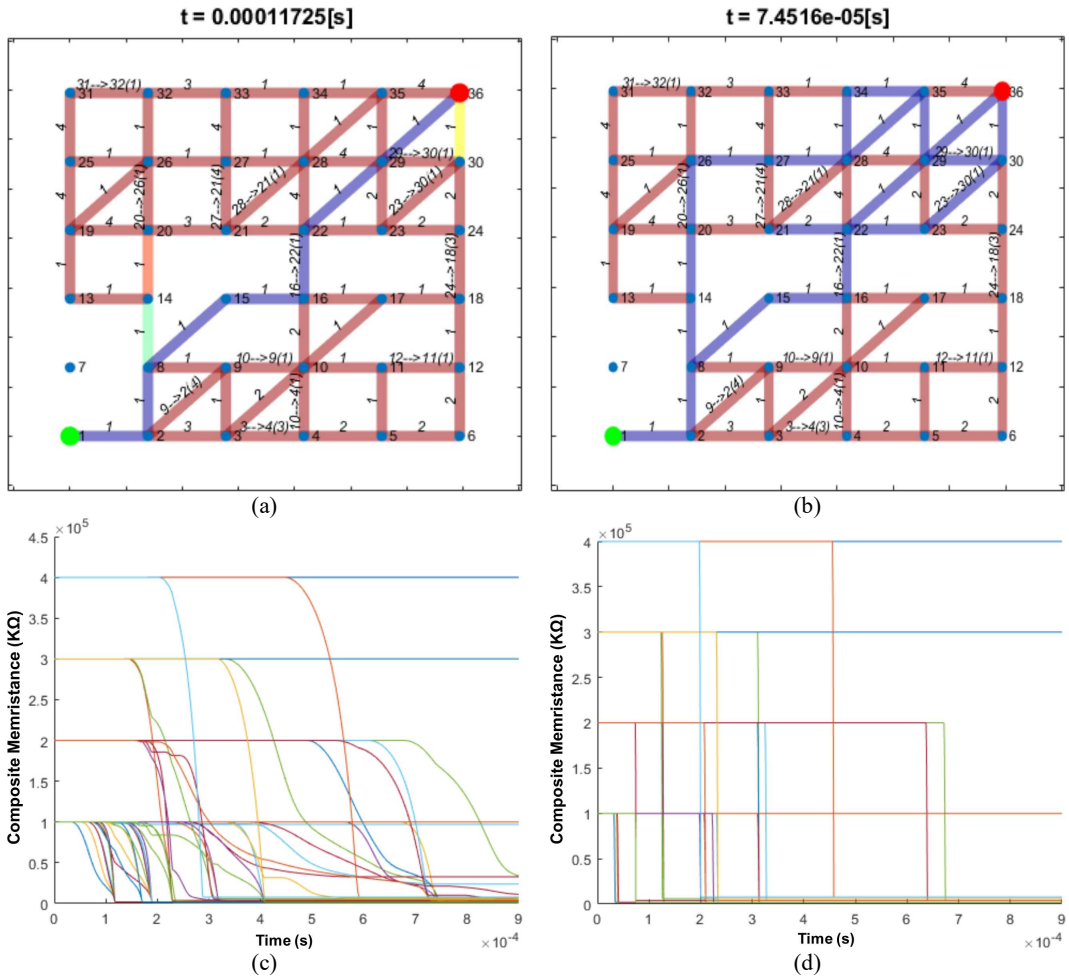
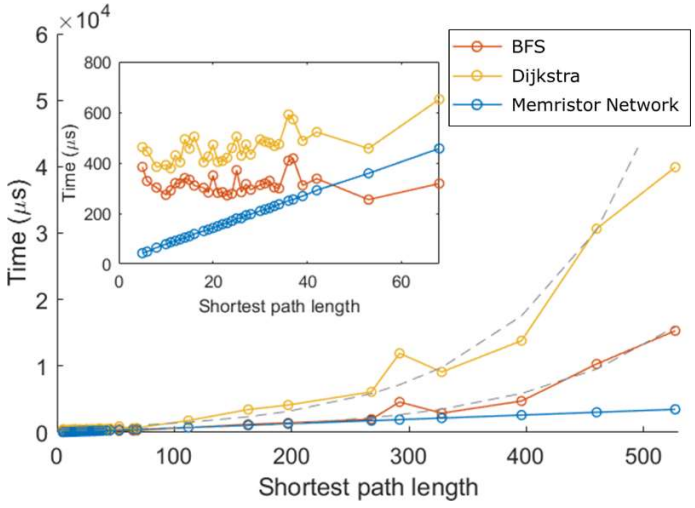


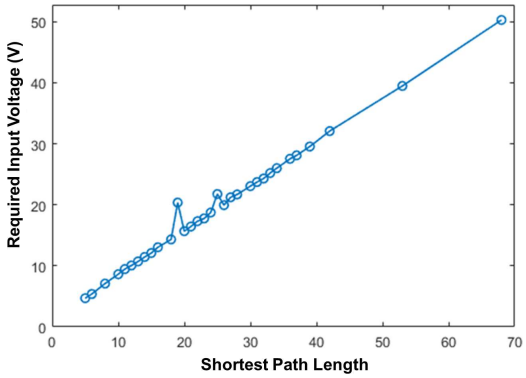
Fig. 7 Visualization of the simulation results for av SPP computation from node 1 to 36, in a random weighted directed graph on a 6×6 grid, during the application of a voltage ramp that rises from 0 to 100V in 1ms, using (a) the model of Pershin [19] with parameters $[R_{ON}, R_{OFF}] = [2, 200]$KΩ, $\alpha = 0$, $\beta = 0.001$e12, and $vt = 1$V; (b) the binary resistive switch model with the same threshold value and resistive range. (c) and (d) provide the time evolution of the composite memristance of all graph edges, corresponding to (a) and (b), respectively.

5. Comparison with Conventional Algorithmic Solutions to the SPP

In conventional digital computers, complicated tasks such as solving a linear system or solving a differential equation require a large number of computing steps and an extensive use of memory. However, to accelerate the execution of such advanced tasks, *in-memory computing* with resistive memory provides a promising solution, owing to the compact multi-level data storage and the physical computation realized in memory. For instance, it was recently shown in [28], [29] that matrix-vector multiplication can be performed with $O(1)$ complexity using a crossbar resistive memory where the matrix values have been mapped linearly to the conductance of the memristive cross-points, thus proving a significant speed-up over the polynomial time complexity of classical digital computers.



(a)



(b)

Fig. 8 (a) Comparison between the avg. SPP computation time required by the memristor-network simulated in LTSPICE (using the model of Pershin [19] with the same parameters mentioned in Fig. 7), and two SPP algorithms implemented in Matlab©, for random weighted directed graphs of variable shortest path length defined in a ###×### grid. The inset shows enlarged the bottom-left part of the plot. Dashed lines correspond to fitting curves to the BFS and Dijkstra data, whereas memristor network data for solution lengths > 70 are result of extrapolation. (b) Evolution of the avg. minimum input voltage required by the memristor network versus the length of the shortest path, for 32 different path lengths in a large number of random graphs defined in a ###×### grid.

Likewise, memristor network-based processing could enable massively parallel analog computing hardware for the accelerated solution of complex optimization problems [30]. In this context, it was recently proven that the *computation time correlates strongly with the length of the shortest path* in the graph, and that this scaling does not depend on the topology of the graph [18]. This is a great advantage of such computing platform, while known algorithmic methods typically scale with the size of the graph (number of nodes + number of edges). We performed several simulations with large random weighted directed graphs where we modified the shortest path length (measured in number of edges), comparing the memristor network performance obtained through circuit simulations in LTSPICE, with that of the Breadth First Search (BFS) and the Dijkstra algorithms, readily available in Matlab©. The results are summarized in Fig. 8 where we show the required computation time

against the shortest path length. The inset presents enlarged the bottom-left part of the plot. We observe clearly the linear dependence of the computation time of the memristor network on the solution length. BFS and Dijkstra algorithms were executed 1000 times for each solution length and the average time was computed, so as to minimize dependency of results on the actual workload of the computer at the moment the simulations were carried out. We simulated cases with maximum solution length longer than 500 in weighted directed graphs mapped in a ###×### grid. For circuit simulations, the computation time considered was the simulated time and not the real duration of the simulations, as the latter depends on the LTSPICE engine; after all, the objective is to compare the time a real memristor network would take to realize the computation.

Our results prove that the memristor network-based approach could be particularly efficient for large graphs with small shortest paths. The complexity of BFS is $O(E+N)$ (N = number of nodes, E = number of edges) whereas that of Dijkstra is in our case $O(E*\log N)$. On the contrary, complexity of the memristor network solution is $O(E_{SP})$, where E_{SP} concerns the subset of the total edges which belong to the shortest path. Consequently, the memristor network based approach is asymptotically more efficient assuming that $E_{SP} << E$.

This notwithstanding, the shortest path length is not known a priori, thus a ramped input voltage is necessary until the minimum required amplitude is reached to get a solution. This method, however, implies an *idle time interval* at the beginning of the computation while the rising input voltage is still low. Certainly, the total computation time required by the memristor network could be drastically improved by reducing this idle time. Fig. 8(b) demonstrates the correlation obtained from several simulations between the required input voltage amplitude and the length of the shortest path (which is in line with results in [18]). Given this information, if the solution length is approximately known, then adding an offset to the input ramped voltage according to an "expected" shortest path length will help reaching faster the required amplitude and thus to reduce the total computation time, making memristor-network based computations further competitive to conventional algorithmic solutions.

6. Conclusions

This work assessed the performance of memristor-network based graph solvers, particularly focusing on the shortest path computation in weighted directed graphs. Conclusions concern the desired memristor device behavior and the impact on computation time, the viability of a previously proposed criterion for the detection of a solution, as well as on the precise read-out of the devices on the shortest path. The impact of analog vs binary switching was evaluated. The memristor-network based computations are completed in a really short time, owing to the physical computing with Ohm's and Kirchhoff's laws. Our simulation results using μs-switching memristors, proved that the memristor networks are advantageous over conventional algorithmic solutions, thus strongly supporting resistive computing as a promising approach for the accelerated solution of complex optimization problems, as well as data analysis and machine learning.

Acknowledgment

This work was supported by the Chilean research grants CONICYT REDES ETAPA INICIAL Convocatoria 2017 No. REDI170604, CONICYT BASAL FB0008, and by the Spanish MINECO and ERDF (TEC2016-75151-C3-2-R).

Declaration of Interest

None.

References

[1] J. Gomez, I. Vourkas, and A. Abusleme, "Exploring Memristor Multi-Level Tuning Dependencies on the Applied Pulse Properties via a Low Cost Instrumentation Setup", *IEEE Access*, vol. 7, pp. no. 1, pp. 59413 – 59421, 2019

[2] Knowm Inc. Neuromemristive Artificial Intelligence. [Online]. Available: https://knowm.org [Accessed: Oct. 30, 2019]

[3] Y. Zhang, X. Wang, Y. Li, and E. G. Friedman, "Memristive Model for Synaptic Circuits," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 64, no. 7, pp. 767-771, 2017

[4] L. V. Gambuzza *et. al.*, "Memristor crossbar for adaptive synchronization," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 8, pp. 2124-2133, Aug. 2017

[5] MemComputing Inc., "Powerful Co-Processors with Speed like no other," [Online]. Available: http://memcpu.com [Accessed Oct. 30, 2019]

[6] M. Di Ventra and Y. V. Pershin, ''The parallel approach,'' *Nature Phys*., vol. 9, pp. 200–202, Apr. 2013

[7] S. Stathopoulos, *et al*., "Multibit memory operation of metal-oxide bi-layer memristors," *Scientific Reports*, vol. 7, no. 17532, 13 Dec. 2017

[8] H. Li, T. F. Wu, S. Mitra, and H.-S. P. Wong, "Resistive RAM-centric computing: Design and modeling methodology," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2263–2273, Sep. 2017

[9] C. Li, *et al*., ''Analogue signal and image processing with large memristor crossbars,'' *Nature Electronics*, vol. 1, pp. 52–59, 2018

[10] F. Sheldon, P. Cicotti, F. Traversa, and M. Di Ventra, ''Stress-Testing Memcomputing on Hard Combinatorial Optimization Problems,'' *IEEE Trans. on Neural Networks and Learning Systems* (2019), *in press*

[11] M. A. Nugent, T. W. Molter, "Thermodynamic-RAM technology stack," *Int. Journal of Parallel, Emergent and Distributed Systems*, vol. 33, no. 4, pp. 430-444, 2018

[12] C. Nakagaki, H. Yamada, and A. Toth, ''Maze-solving by an amoeboid organism,'' *Nature*, vol. 407, no. 470, 28 September 2000

[13] Y. V. Pershin and M. Di Ventra, ''Self-organization and solution of shortest-path optimization problems with memristive networks,'' *Phys. Rev. E*, vol. 88, no. 1, p. 013305, Jul. 2013

[14] I. Vourkas, D. Stathis, and G. Ch. Sirakoulis, ''Massively Parallel Analog Computing: Ariadne's Thread Was Made of Memristors,'' *IEEE Trans. on Emerging Topics in Computing*, vol. 6, no. 1, pp. 145-155, Jan.-March 2018

[15] I. Vourkas and G. Ch. Sirakoulis, “Networks of memristors and memristive components,” in *Memristor-Based Nanoelectronic Computing Circuits and Architectures*, 1st ed., Switzerland: Springer Int. Publishing, 2016, pp 173-198
[16] Z. Ye, S. H. M. Wu, and T. Prodromakis, “Computing Shortest Paths in 2D and 3D Memristive Networks,” In: Adamatzky A., Chua L. (eds) *Memristor Networks*, Springer, Cham, 2014, pp. 537-552
[17] C. Fernandez and I. Vourkas, “Performance Assessment of Memristor Networks as Shortest Path Problem Solvers,” *unpublished*
[18] A. Mizrahi, T. Marsh, B. Hoskins, and M. D. Stiles, ‘‘Scalable Method to Find the Shortest Path in a Graph with Circuits of Memristors,’’ *Phys. Rev. Applied*, vol. 10, no. 064035, 14 December 2018
[19] Y. Pershin and M. Di Ventra “SPICE Model of Memristive Devices with Threshold,” *Radioengineering*, vol. 22, no. 2, pp. 485–489, 2013
[20] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, “Memristor SPICE Modeling,” in: Kozma R., Pino R. E., Pazienza G. E. (Eds.) *Advances in Neuromorphic Memristor Science and Applications*, Springer Series in Cognitive and Neural Systems, vol 4. Springer, Dordrecht, 2012, pp. 211-244
[21] A. Ascoli, F. Corinto, V. Senger, and R. Tetzlaff, “Memristor model comparison,” *IEEE Circ. Syst. Mag.*, vol. 13, no. 2, pp. 89–105, 2013
[22] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, “The missing memristor found,” *Nature*, vol. 453, pp. 80 - 83, 2008
[23] Z. Biolek, D. Biolek, and V. Biolkova, “SPICE model of memristor with nonlinear dopant drift,” *Radioengineering*, vol. 18, no. 2, pp. 210–214, 2009
[24] P. S. Georgiou, S. N. Yaliraki, E. M. Drakakis, and M. Barahona, “Window functions and sigmoidal behaviour of memristive systems,” *Int. J. Circ. Theor. Appl.*, vol. 44, np. 9, pp. 1685–1696, 2016
[25] A. Nugent, and T. Molter, “AHaH Computing–From Metastable Switches to Attractors to Machine Learning,” *PLoS ONE*, vol. 9, no. 2 (e85175), February 10, 2014
[26] Memristor Models in LTSpice. [Online]. Available: https://knowm.org/memristor-models-in-ltspice/ [Accessed: Dic. 08, 2019]
[27] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, “Generalized Memristive Device SPICE Model and its Application in Circuit Design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 8, pp. 1201-1214, 2013
[28] A. Sebastian, *et al.*, “Tutorial: Brain-inspired computing using phase-change memory devices,” *Journal of Applied Physics*, vol. 124, no. 11, pp. 111101, 2018
[29] Z. Sun, *et al.*, “Solving matrix equations in one step with cross-point resistive arrays,” *PNAS*, vol. 116, no. 10, pp. 4123-4128, 2019
[30] Memryx: Reconfigurable in-memory computing chips with unparalleled energy efficiency and compute density. [Online]. Available: https://www.memryx.com [Accessed: Oct. 30, 2019]