Abstract—This paper explores memristive grids where emergent computation arises through collective device interactions. Computing efficiency of the grids is studied in several scenarios and new composite memristive structures are utilized in shortest path and maze-solving computations. The dependence of the computing medium behavior on the symmetry of both the underlying geometry and the employed devices is validated through SPICE-level circuit simulations, which highlight important computing inefficiencies. Particular circuit-models of memristive connections enable precise mapping of the target application on the computing medium. Extraordinary functionalities emerge when novel memristive computing components, comprising different electrical characteristics from their structural elements, are introduced in the grid. Applying assisted-computation, by incorporating the concept of Ariadne’s thread, led to better computing results, which could find application in routing and path computing problems.

Index Terms—memristor, memristive grid, programmable circuits, computing, shortest path, maze solving

1 INTRODUCTION

Back in the early days of digital circuits, analog computations embodied a whole area of research which, however, was not as scalable and/or reproducible as former digital solutions. This is one of the reasons why analog computing was not given much of attention afterwards. Nevertheless, nowadays there are very important areas of engineering and modeling problems which still require analog-based computational methods and structures [1]. In the same context, for decades researchers were convinced that real artificial intelligence (AI) would never be done on conventional hardware (HW), where processing and memory units are strictly separated [2]. However, not until when Hewlett-Packard (HP) Labs built the first memristor prototype in 2008, it was believed that creating something with the function principles of a brain could finally be possible [3-5].

Memristors demonstrate a natural basis for computation which combines information processing and storage in the memory itself. Owing to their analog memory functionality, an extraordinary type of computing parallelism was introduced in [6], dubbed as memcomputing. This type of analog parallelism consists in array-like structures which accommodate large numbers of memristors (or generally memelements [7]) where complex, unconventional, and/or neuromorphic computations take place. The strength of memcomputing is essentially based on the massively-parallel analog dynamics of many interconnected memristors and the ability to recover the result of the computation from the same computing units, much like the brain is thought to operate. The development of computational architectures and methods based on memristors constitutes a large area of ongoing research with examples which extend from unconventional logic design to Cellular Neural Networks (CNN) and neuromorphic computing with memristive synapses [8-13]. The latter, aims to use biological mechanisms within the brain as a blueprint for novel computer architectures [14-18], and is considered one of the most promising applications of memristors [19]. There are, though, complex challenges that can be solved with the help of memristors, but without necessarily mimicking the synaptic functionality. A very powerful computing structure shown here, which implements analog parallel computations, is the so called memristive grid (network).

In such a structure, there is continuous information exchange during calculations, which renders a tremendous increase of computational power due to the massively-parallel network dynamics. Calculation consists in the evolution of the states of all the involved devices. In spite of being still at an early stage, related work in memristive grid-based computing has shown promising results in graph theory optimization problems [20-22] and signal processing [23, 24]. However, most of these research approaches focus on homogenous networks which mainly consist of ensembles of identical and reciprocal (either parallel or in-series) memristors [25], [26] in hopes of achieving structural simplicity. As a consequence of the symmetry of interconnections, though, such type of grid sometimes fails to provide a unique solution to each problem since it hardly converges to an easily distinguishable steady state during computation. It is the same symmetry that also impedes the appropriate mapping (projection) of the target problem-space (more generally a target application described in the form of a directed graph) on the computing medium.

In this work, we address two of the most well-known and well-documented, inherently complex (in terms of computation time) problems, i.e. the shortest path and the maze-solving problems, via computations in memristive...
2 MEMRISTIVE GRID-BASED COMPUTATIONS

2.1 Background on Memristors

Memristor, the fourth circuit element (joining the resistor, the capacitor, and the inductor) predicted by Chua back in 1971 [27], represents one of today’s latest technological achievements. Its unusual electronic behavior and rich dynamics made it one of the most promising candidates beyond conventional CMOS technology. Memristor (here used to describe both an ideal memristor as well as a generalized memristive system [28]) is a passive two-terminal electronic device whose behavior is described by a nonlinear constitutive relation:

\[ v = R(x,i)i \]  

between the voltage drop at its terminals \( v \) and the current flowing through the device \( i \), where:

\[ \dot{x} = f(x,i) \]  

where \( x = (x_1, x_2, ..., x_n) \) denotes \( n \) state-variables \( x_1, x_2, ..., x_n \) which do not depend on any external voltages or currents, whereas the nonlinear function \( R(x,i) \) is called memory resistance (memristance) and has the unit of Ohms (\( \Omega \)). The reason why memristors are substantially different from the other fundamental circuit elements is that, when the applied voltage is turned off, they still remember how much voltage was applied before and for how long; thus presenting memory of their past. That’s an effect that cannot be duplicated by any circuit combination of resistors, capacitors, and inductors, which qualifies the memristor as a fundamental circuit component. The nonvolatile memory property of memristors is a direct consequence of the state-dependent Ohm’s law in (1).

Threshold-type switching is closer to the actual behavior of most experimentally realizable memristive devices. Throughout this work, all conducted simulations employ a threshold-based device model of a voltage-controlled memristor which attributes its behavior to a tunneling distance modulation [8]. Such model is based on the assumption that the resistance switching-rate of a memristor is small below (fast above) a voltage threshold (namely \( V_{SET} \) or \( V_{RESET} \)), which is viewed as the minimum voltage required to impose a change on the physical structure (and thus the memristance) of the device. The model permits assuming asymmetric thresholds \( \{ |V_{SET}| \neq |V_{RESET}| \} \) and different tunneling distance change-rates for the SET and RESET operations. A PSPICE version of it was published in [29]. The simulation results qualitatively and quantitatively match the theoretical formulation. The values of the parameters of the model are set as given in [29] with \( \{ R_{ON}, R_{OFF} \} = [2, 200] \text{k}\Omega \) and \( \{ V_{SET}, V_{RESET} \} = [2, -1] \text{V} \).

2.2 Computing Platform Description

 Appropriately interconnected memristors significantly improve the efficiency of computations via massive parallelism. In its more generic form, the assumed memristive

grids. Solution of the shortest path problem (SPP) has always been a hot topic in graph theory because of its wide application field. The inherent difficulty in this problem stems from the fact that any two points of the plane are often connected by multiple degenerate paths, which complicates the finding of a single optimal solution. On the other hand, maze-solving is a tour puzzle, i.e. a complex arrangement of pathways in which the correct path must be found. Mazes have fascinated people since the ancient times. According to an ancient Greek myth, Ariadne gave a ball of thread to Theseus so that he could find his way out of the Minotaur’s labyrinth. Ariadne’s thread, named after the heroine of the myth, is nowadays how we call the solving approach of a problem with multiple apparent means of proceeding through exhaustive application of logic to all available routes.

Inspired by the aforementioned myth, here we show how the computing grid could be guided in order to provide better solutions by laying a memristive thread across several grid-nodes. We further extend already proposed solving approaches to the aforementioned problems by introducing certain modifications in the computing platform, which here may comprise sophisticated memristive structures other than just reciprocal devices. Also, we provide circuit models corresponding to several types of connections between graph vertices, thus enabling the precise network projection of any mesh-based directed graph with \( n \) vertices arranged on a discrete lattice with nonnegative connection weights. Several scenarios are examined considering also the inclusion of devices with different switching characteristics in the same computation, thus getting past the problems caused by network symmetry. The dynamically changing state of this adaptable medium, in response to time-dependent signals or changes in the grid configuration, is monitored and thoroughly discussed. Explicit computing simulation examples, using either a developed memristive network simulator or PSPICE simulation environment, provide intuition into the capabilities and the weaknesses of this new class of computing HW. Moreover, the emergence of unforeseen functionalities opens doors to exciting new computing concepts and encourages the development of parallel memristive computing systems which could have a large impact in several practical problems. Finally, the presented assisted-evolution of the memristive grid raises the point that the famous thread of Ariadne was made of memristors!

The rest of the paper is organized as follows. Section 2 provides a brief background on memristors, describes the operation principles of massively-parallel grid-based computations, and explains the application mapping rules. Section 3 includes a simulation-based study of shortest path and maze-solving computations, incorporating the concept of Ariadne’s thread, and discusses novel functionalities by using composite memristive computing components. Section 4 concludes the paper.
grid comprises a mesh of nodes which are connected by computing components involving bipolar memristors, resistors, and/or switches (transistors); the latter have two roles: (i) they facilitate mapping of the target problem-space onto the HW network when the latter is regular and homogenous as in [20]; (ii) they provide independent access to individual memristors for the purpose of programming or reading of their state. At present it is inevitable that such dynamical circuits must be interfaced with conventional electronics to produce and observe signals. Therefore, it is assumed that external signals can be applied to any grid nodes for the purpose of initialising the memristors’ states, as well as to read the calculation results by a memristance read-out of all memristors.

Overall, the massively-parallel operation of such platform consists in three main stages: (i) initialization; (ii) computation; and (iii) reading of the outcome. The computation stage normally consists in the application of voltage pulses of appropriate amplitude and duration across specific nodes of the grid, while grounding others. Computation is initiated by triggering the source nodes and then a wave of stimulation propagates in all directions and affects accordingly the states of the rest of the devices. In any moment, the potential at all grid points can be found as a solution of Kirchhoff’s current law (KCL) equations. The time required for the grid to reach a steady state during calculations depends on the applied signals, the network topology, and the switching characteristics of the individual memristors. The computation result is given by a sub-set of components, which are found in a predefined resistive state. Normally, all memristors are initially set in the high resistive state ($R_{OFF}$), so the network marks the solution with the lowest corresponding memristances (or intermediate states if memristors did not have enough time or voltage across them to completely switch state) while it successively finds all possible solutions.

Fig. 1a shows an indicative snapshot of a regular memristive computing grid. Between vertically and horizontally neighboring nodes there is a memristive circuit which can vary according to the target application as shown in Fig. 1b. Depending on the nature of the problem to be solved, the direction of current flow in the grid is not always known a priori. However, if current direction is important, then single forward-polarized memristors, whose polarity is in line with the desired current flow, are suitable for modeling such connections (their memristance will decrease when forward-biased). Otherwise, a pair of series/parallel memristors with opposite polarity is used; thus, the symmetry of the memristive compositions provides operation-independency on the sign of the voltage at their terminals. Moreover, Fig. 1c shows two possible ways of creating memristive devices with higher switching thresholds, to be used in the grid. In fact, higher thresholds could be achieved either by incorporating pairs of anti-parallel diodes in series with the memristors [30], or by following the concept of composite memristive structures presented in [31]. The latter consists in using small network configurations of memristors which, as a whole, it demonstrates the same overall memristance range [$R_{ON}$, $R_{OFF}$] with single memristors, but can have $n$-fold ($n \times$) cumulative switching thresholds.

### 2.3 Circuit Models for Application Mapping

Given the variety of possible interconnections found within a target application graph with $n$ vertices, which
are arranged on a discrete lattice with nonnegative link-weights, in Fig. 2 we propose a list of corresponding circuit modeling approaches (the transistors in series to all memristors are omitted for simplicity). When a given graph is first mapped onto the network, connections between the vertices can be either unidirectional, bidirectional, or completely closed. By following the provided list of options, any kind of directed graph whose edges are of equal weight, can be easily projected. Unequal weights could be modeled, as well, by using memristors with different switching characteristics, e.g. different memristance ratio and/or voltage thresholds.

According to Fig. 2, typical resistors are used: (a) to model connections which can be by default included or excluded from the solution (e.g. by setting their value equal to the less resistive or the high resistive state of the used memristive components); (g) to represent open circuits (the resistor has much higher resistance than the higher equivalent resistance of a memrist or a memristive composition); (f) or short circuits (very small resistance). Moreover, the series/parallel coupling of memristors with resistors to model unidirectional links in (b) and (c) should be selected according to the choice made for the bidirectional links in (d) in order to maintain a common memristance range for every memristive connection between the nodes of the grid. For example, if anti-parallel memristors are selected from (d) for the bidirectional links, then their memristance varies within $[R_{\text{ON}} || R_{\text{OFF}}, \frac{1}{2} \times R_{\text{OFF}}]$; this exact memristance range also corresponds to the parallel connection of a memristor with a resistor whose resistance is equal to $R_{\text{OFF}}$ (operator $||$ denotes parallel connection of two resistive elements). For directed graphs with purely unidirectional links single memristors would do fine, whereas for purely bidirectional connections we choose only reciprocal in-series/parallel memristors. The choice between the latter impacts the required input voltages since series elements require higher voltages to switch their states due to the voltage divider. Case (e) is a special type of memristive component combining a pair of anti-parallel with a pair of anti-series memristors. Its use will be further explained in the following sections.

3 Path Computing and Maze-Solving with Ariadne’s Memristive Thread

Next we study the behavior of the grid in several scenarios and experiment with the presented composite memristive structures, as well as with different switching properties of the devices. We show how, what appears to be a competition for the available applied voltage across the interconnected devices, can be exploited in shortest path and maze-solving computations. For the purpose of this study we developed a memristive network simulator using the Easy Java Simulations (EJS) environment [32] where all differential equations of the employed threshold-type memristor model [8] are numerically solved using a 4th order Runge-Kutta integration method. For better visualization of the induced memristance change to each composite computing component, a linear color scale is used to represent all possible equivalent memristance values. Given the known precise quantitative match of the SPICE version of the employed model, we also selectively performed circuit simulations using the Cadence PSPICE environment in order to validate the outcome of our computations.

3.1 Fully Interconnected Network

In order to demonstrate the fundamental principles of operation of the memristive grid, we simulated a 9×9 regular mesh where we first considered the solution of the SPP along various directions between two (or more) pre-selected nodes. In our first demonstration, we assume a fully interconnected undirected mesh; therefore, between every pair of neighboring grid nodes we place two identical anti-parallel memristors, initially set in $R_{\text{OFF}}$. A typical computation is initiated by applying a voltage pulse of a suitable amplitude and duration to the source node(s) while grounding the destination node(s). The sub-set of components that first switch to the less possible resistive state ($R_{\text{ON}} || R_{\text{OFF}}$) gives the result of computation.

Fig. 3 shows the simulation results of the first case; Fig. 3a provides the visualization of the network evolution during the calculation stage by focusing on three distinct time points where up to three different solutions are given. The grid nodes are colored black and the memristance range of all the devices employed in computation is given according to the provided color map ($R_{\text{MIN}}$ and $R_{\text{MAX}}$ denote the min and max composite resistance of each memristive connection). Fig. 3b illustrates the current-time ($i$-$t$) plot of the currents flowing through all of the 144 memristive components (i.e. pairs of anti-parallel memristors) involved in computation, taken from PSPICE. The vertical dashed lines denote the time moments corresponding to the snapshots shown in Fig. 3a. Our results are in very good qualitative agreement with
those found in the literature [22]; the expected shortest path is the first to emerge during computation. Two alternate equivalent solutions appear simultaneously after the best solution has been calculated (negative measured currents are due to a typical SPICE convention for the current flowing inwards or outwards of a particular device, depending on which terminal the ammeter is placed).

In such solutions the total resistance of every path is proportional to its length. The network has no external clock and finds all the solutions in parallel, although the final read-out requires subsequent resistance measurements. If there are multiple paths, then the shortest one contains the fewer memristors and, thus, evokes less resistance than the longer ones. All intermediate paths (in terms of length) offer a proportionate resistance. Therefore, since current flows in inverse proportion to the resistance of a path, the change of state of a given path is proportional to the current in the path. Even more paths are revealed as computation continues in time (it is not shown in this example); alternate shortest paths can be identified by the different state of the respective memristive connections. However, multi-state reading with memristors is difficult, so it is preferable to distinguish between fully-switched devices other than devices that are in intermediate resistive states. Between memristive components from two possible paths, those of the shorter path switch their state at an earlier moment than those of the longer path. This allows sorting all possible solutions according to their length. Since all neighbor connections in the network are considered to be of equal weight, the aggregate length of a particular path is here identified by the total number of “hops” in-between source and destination nodes.

After a definite time, almost all the memristive components are expected to finally switch completely or reach an intermediate state, which will possibly ruin the reading process afterwards. Of course, less switching events are expected if the memristors are supposed to be affected exclusively by voltages which exceed their thresholds. In this example, the source node for the circuit simulation is exclusively by voltages which exceed their thresholds. In this example, the source node for the circuit simulation is represented by a DC voltage source whose amplitude is set high enough so as to exceed the accumulated threshold value of the amount of devices belonging to the easily observed shortest path. However, when the source/destination nodes are located along an arbitrary direction, different from the network symmetry directions (i.e. that cannot be vertically or horizontally connected directly), the minimum necessary voltage amplitude, which will shortly lead computation to a unique solution, is hard to be estimated. Therefore, it is preferable to apply a slowly ramped waveform voltage while searching for the proper amplitude. This type of input voltage is used in the rest of presented simulation scenarios.

Let us now consider the behavior of the grid in more sophisticated shortest path computations, either when trying to move diagonally or when trying to get past an obstacle in order to reach a destination node (in other words, when the network is damaged). Fig. 4a depicts the evolution of computation of the shortest path going from node (row, column) = (5, 1) to node (9, 3). Two intermediate computing stages are shown, as well as the final steady state, which the grid reaches after a stipulated amount of time. Fig. 4b similarly illustrates the current-time (i-t) plot of all measured currents during circuit simulation using PSPICE, where the vertical dashed line denotes the moment when the network approaches to a steady state, as in the final schematic of Fig. 4a. However, unlike the previous example which had a straightforward solution, here many paths emerge almost simultaneously, thus making it hard to distinguish a particular optimal solution. The steady state of the grid consists in all possible shortest path combinations within the 5-by-3 rectangle located in the bottom-left corner of the mesh. Therefore we can conclude that, when the selected nodes are located along an arbitrary direction within a regular grid (i.e. when row\textsubscript{SOURCE} ≠ row\textsubscript{DESTINATION} and column\textsubscript{SOURCE} ≠ column\textsubscript{DESTINATION}), this computing platform fails to uniquely solve the problem because of its inherent symmetry of interconnections and of the employed devices. We remind here that the simulations are based on voltage-controlled threshold-type switching memristors. Because of this, the result of Fig. 4a diverges from similar published simulation results in the literature, which were based on current-controlled bipolar memristive devices. Specifically, in [22] the authors observed a self-reinforcement of the shortest path solution with time due to the memristive network dynamics; the least resistive path distinguished faster by attracting more and more current. However, in order to achieve that, it was as-
assumed that the rate of memristance ($M$) change was proportional to the current flowing through the devices \([3],[33]\) as shown below:

\[
\frac{dM_{ij}}{dt} = \alpha \cdot I_{ij}(t)
\]

where $\alpha$ is a constant and $I_{ij}(t)$ is the current flowing through the memristive connection \((ij)\). Unless (3) is complied, such dynamics are unfortunately retained.

Fig. 4c presents a variation of the same scenario where, inspired by the famous Greek myth, we tried to lead the path computation through particular intermediate points by spreading a memristive thread along the desired path. More specifically, we chose to exploit network heterogeneity by employing devices with different switching characteristics in the same computation. We particularly assigned lower switching thresholds for some selected memristors (they are highlighted in the first snapshot) which are found in an alternate path that we wish to follow. The simulation shows that the first clear emerging solution appears faster than before (only after $t_1$) and follows the desired path in a zig-zag form. However, as computation continues in time, more paths appear gradually and the steady state of the grid finally coincides with that of Fig. 4a.

### 3.2 Defective Network

In the next scenario, a few connections of the mesh were removed (i.e. replaced with $R_{\text{MAX}}$ resistors) intentionally in order to test the stability of the shortest path problem solution around a defective region of the grid (i.e. an obstacle). According to simulation results in Fig. 5a, the computation evidently tends to reach the shortest possible path around the damaged area. We note here that the memristive network has a remarkable ability to repair damaged solutions. Indeed, this property is close to the self-healing ability that can be ascribed to systems or processes which, by nature or by design, tend to correct any disturbances. The missing connections have been removed intentionally in an asymmetric fashion in order to show that the healing occurs along the shortest possible path around the damaged region. However, as Fig. 5b confirms, again multiple paths emerge almost simultaneously, making it hard to follow the optimal solution. Within a particular simulation time, all possible shortest path combinations that pass from the most convenient side of the defective area are indicated.

Likewise in the previous case, we again spread memristive components which comprise devices with lower switching thresholds within the grid, aiming to guide the platform to a specific solution. As shown in Fig. 5c, although the few adjusted devices are located in the less convenient side of the obstacle, the computation obviously follows the desired path passing across all the
specified connections. Unlike in Fig. 5a, here we observe that the grid converges much faster to the solution and, most importantly, retains this solution for a much longer time before more degenerate paths appear.

### 3.3 Maze-Solving

In Fig. 6 we examine the delineation of a given maze with memristors according to the early approach proposed in [20]. The maze and the possible pathways are shown in Fig. 6a. The maze is then superimposed to a memristive network as shown in Fig. 6b. This particular underlying network comprises a head-to-head and tail-to-tail checkerboard pattern of memristors, which are connected to others via a switch at their tail. The crossing points of vertical and horizontal lines define grid points of the memristive network. In order to encode the pattern, the switch is closed for possible pathways and opened for blocked passages. This leads to a network configuration as shown in Fig. 6c. Larger mazes are prepared in the same way and voltage is applied across the desired entrance and exit points to solve it.

Since the direction of current flow in the network is not known a priori, the polarity of adjacent memristors was chosen to be alternating. According to [20], such architecture allows modeling different mazes on the same memristive grid without the need to fabricate a specific network for each maze. For the memristive dynamics, however, it was again assumed that the rate of memristance change is proportional to the current flowing through the devices, as shown (3). Moreover, the sign of the applied input voltage was changed during simulation in order to better represent the maze solution. In fact, if we change the sign of the applied voltage after some time, then the resistance of memristors—along the solution path—that are in the ON state will increase towards the OFF state, whereas the resistance of memristors that are in the OFF state (which were not affected by the first positive voltage due to being reversely polarized) will now decrease towards the ON state. As a result, at a specific moment of time, every memristor along the solution path is found in the same intermediate state. Such an approach, however, implies that multi-state reading of memristors is possible.

In our case, by replacing certain memristive components of the fully interconnected mesh grid with $R_{\text{MAX}}$ resistors (see Fig. 2), it was easy to map a particular maze on the same structure and experiment with maze-solving. In fact, we further extended the approach of [20] to show how a mesh of memristive components can be utilized for even more sophisticated computations. Entrance and exit nodes for the circuit simulation are represented by an increasing ramp-waveform voltage and ground, respectively. Fig. 7a shows the time evolution of computation when employing identical anti-parallel memristors as computing components for the undirected links. This polarity-independent memristor combination makes unnecessary the change of the sign of the applied voltage since the solution will finally comprise a subset of connections, which will have completely switched state towards the less resistive composite state (i.e. $R_{\text{ON}}$ | $R_{\text{OFF}}$). The results are in absolute qualitative agreement with those of [20]. The shortest solution emerges first whereas other existing paths, which are longer than the first one, are revealed afterwards as possible alternative options; the steady state of the memristive grid includes all possible ways to the exit of the maze without any distinctions. We note that, unlike the fully interconnected mesh grid and the SPP, in maze-solving we have always observed the system to evolve towards a unique (for a given maze) stable solution. Therefore, the success rate of correct solution is 100%.

In the next example we worked on the same maze where we selectively modified the switching thresholds of a few memristive components, found in the non-optimal solution which appeared last in Fig. 7a. In the simulation result of Fig. 7b we notice that the desired solution is the one which now emerges first. Nevertheless, by the end of computation again all available paths are present with no differentiation between them. After a considerable amount of similar experiments with various types of maze, we figured out that, in order to force the grid to converge faster to a non-optimal solution, the total number of remaining memristive components with the high thresholds in the alternate path have to be less in number than those which belong to the originally shortest solution; i.e. that of Fig. 7a. Therefore, the longer the thread of Ariadne is, the higher the chances are for the grid to reach a desired solution. Of course, in this assisted-type of network evolution, the same result occurs if we place resistors instead of memristors with lower thresholds; only that
these resistors should have a resistance equal to \((R_{\text{ON}} \parallel R_{\text{OFF}})\) in order to be a priori included in the solution.

**Composite structures enable spotting closed loops**

As mentioned before, the emergence of unforeseen functionalities could pave the way to exciting new computing concepts via memristive networks. To this end in Fig. 8 we present the simulation results for three different maze-like network structures. Unlike in previous examples, here we used the circuit of Fig. 2e to model network connections. In this composite structure, an additional pair of anti-serial memristors is found in series with the anti-parallel memristors which were used so far. These two additional devices are initially set in the less resistive state \((R_{\text{OFF}})\) and their thresholds are selected higher than those of the anti-parallel memristors. Hence this new configuration will not affect significantly the previously presented computations since the equivalent resistance of this composition is only slightly higher than before; in fact the initial resistance is equal to \(\frac{1}{2}R_{\text{OFF}} + 2\times R_{\text{ON}} \approx \frac{1}{2}R_{\text{OFF}},\) since \(R_{\text{OFF}} >> R_{\text{ON}}\).

Under the application of a slowly ramped waveform voltage across the indicated nodes, the evolution of the grid will occur as expected, revealing gradually all possible alternative ways from the start to the end points according to their length. However, as the total applied voltage increases, the corresponding voltage drop on the additional series memristors of each component soon exceeds their threshold. Thus, one of them (depending on the voltage polarity) switches to the high resistive state. As a consequence, the composite memristance rises and becomes equal to \((R_{\text{ON}} \parallel R_{\text{OFF}}) + R_{\text{OFF}} + R_{\text{ON}} \approx R_{\text{OFF}}.\) This way of increasing the composite resistance between certain parts (connections) belonging to the solution sub-set, provides the ability to eventually exclude these parts from the final solution, thus leaving only the components that have switched to the less resistive composite state.

As particularly shown in Fig. 8, computation evolves up to a certain point similarly to when employing only anti-parallel memristors between the grid nodes. Afterwards, as computation goes on and the applied voltage keeps increasing, we demonstrate that this new memristive structure enables the localization of closed loop paths between common points in any undirected graph mapped on a maze-like structure. In all presented cases of Fig. 8, the common parts of the solution gradually disappear (i.e. their composite memristance increases) and computation reaches a steady state where only closed loops are shown. This extraordinary behavior of the grid is due to the fact that the equivalent resistance of any closed loop, i.e. of any two parallel circuit branches between two particular nodes, is smaller than the resistance of the single common parts of a solution. Therefore, the corresponding voltage drop on the single parts results higher and hence exceeds sooner the thresholds of the additional series memristors, causing one of them to finally switch OFF as described before. This novel ability of memristive grids, which utilize composite memristive structures, could find many applications in routing and path planning problems, e.g. for the identification of the critical components in specific routes within congested transportation or communication networks, provided that the latter are appropriately mapped on the computing medium.

**4 Discussion**

Numerical modeling of memristive networks is easily implemented and offers a practical computational algorithm for several known problems, whose solution is hard to be computed fast. Here, both high-level, as well as circuit-level simulations via SPICE, revealed the pros and cons of such unconventional computing approach for SPP and maze-solving applications. Important computing inefficiencies were attributed to the dependence of the computing medium behavior on the symmetry of both the underlying geometry and the employed devices. Extraordinary functionalities emerged when novel memristive computing components were introduced, whereas incorporating the concept of *Ariadne’s thread* allowed interfering in computations.

Experimentally, the suggested networks could be fabricated, e.g. by combining one (or more) memristor layer(s) (or memristor emulators [34]) with an additional complementary metal-oxide-semiconductor (CMOS)
layer(s). The physical production of such complex circuit architectures would be a great achievement because such systems would provide adaptability and computing capabilities at a rate determined by device physics and not by algorithmic complexity. In this context, unlike mentioned in similar studies in the literature [20], [22], the memristive grid does not require only a single step to find the solution. In fact, the evolution of computation depends on the switching characteristics of the employed memristive devices and on the underlying network which, in turn, imposes different requirements for the applied input voltage. Higher applied voltages might cause all solutions to emerge simultaneously and not gradually as it is desired; the minimum necessary voltage amplitude which gives a first solution might be only roughly estimated. Nevertheless, the computation time is independent of the network size and/or complexity.

With respect to path planning problems, these are typically evaluated using four metrics: (i) time complexity (the time needed to find a solution); (ii) space complexity (memory needed for the search); (iii) completeness (a solution is found if it exists); and (iv) optimality (the best solution is found) [35]. Regarding (i), for systems that can search multiple grid nodes in parallel, such as in [17], if we assume equal connection weights then the propagation of the signal is uniform in the grid. This is in-line with the presented parallel solution approach with memristors, which is a priori more time-efficient than any multistep algorithm in present use. Compared to initial neuron training of a neuron IC, which could be time-consuming, the memristive platform only requires a common initialization of all the involved devices. Of course, the state read-out time cannot be avoided, but we believe it is comparable with measurements of current neuromorphic solutions like in [17], and here is certainly compensated by the minimal computation time. Concerning (ii), in our case it is related to both the network size and the node connection type; the more adaptive and robust the platform is made, the larger its size complexity. Finally, requirements (iii) and (iv) are absolutely covered by memristive networks.

Overall, the most important parameters about memristive grid-based computations concern: (i) the size of the grid, which should be large enough to accommodate the target application graph; (ii) the switching time and thresholds of the memristors, which should be known to facilitate the selection of the applied input voltage; (iii) the type of memristors, which should demonstrate threshold-based switching; and (iv) the connection type of the target application graph, which should correspond to the available node-connecting components of the grid.

5 Conclusion
This work presented a study of novel analog computing concepts utilizing massively-parallel computing architectures, enabled by networks of memristors. SPICE-level simulations revealed the pros and cons of such unconventional computing approach for shortest path and maze-solving applications. The incorporation of Ariadne’s thread concept in assisted-computation, using devices with different switching characteristics, revealed better performance of the computing medium and allowed interfering in computations. Extraordinary computing behavior was revealed when sophisticated memristive computing structures were introduced in the grid. The demonstrated novel abilities of memristive grids could find many applications in routing and path planning computations on oriented graphs.

Acknowledgment
This work was supported in part by a scholarship from the Bodossaki Foundation in Greece. Authorship correspondence to: Ioannis Vourkas, Laboratory of Electronics, Department of Electrical and Computer Engineering, Democritus University of Thrace, Panepistimioupoli, Ktimmeria, GR-671 00, Xanthi, Greece.

References
This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.

The final version of record is available at http://dx.doi.org/10.1109/TETC.2015.2420353
several other conferences, including ACRI 2012, the 5th Pan-Hellenic ECE Students Conference (SFMMY 2012), and of special sessions in IEEE NANO 2015, the 23rd Euromicro Int. Conf. on Parallel, Distributed and network-based Processing (PDP 2015), the HiPEAC Computing Systems Week (CSW) 2014, PDP 2014, the 18th Panhellenic Conf. in Informatics (PCI 2014), the 20th IEEE Int. Conf. Electronics, Circuits, and Systems (ICECS 2013), etc.