Received May 2, 2019, accepted June 7, 2019, date of publication July 9, 2019, date of current version August 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927815

Parallel Accelerated Virtual Physarum Lab Based on Cellular Automata Agents

NIKOLAOS I. DOURVAS, GEORGIOS CH. SIRAKOULIS, (Member, IEEE), AND ANDREW I. ADAMATZKY
Department of Electrical & Computer Engineering, Democritus University of Thrace, 671 00 Xanthi, Greece
Department of Computer Science, University of the West of England, Bristol BS16 1QY, U.K.

Corresponding author: Nikolaos I. Dourvas (ndourvas@ee.duth.gr)

This work was supported in part by the General Secretariat for Research and Technology (GSRT), and in part by the Hellenic Foundation for Research and Innovation (HFRI).

ABSTRACT Self-aware and self-expressive physical systems are inspiring new methodologies for engineering solutions of complex computing problems. Among many other examples, the slime mold Physarum Polycephalum exhibits self-awareness and self-expressiveness while adapting to changes in its dynamical environment and solving resource-consuming problems like shortest path, proximity graphs or optimization of transport networks. As such, the modeling of the slime mold's behavior is essential when designing bio-inspired algorithms and hardware prototypes. The goal of this paper is to combine one of the powerful parallel computational tools, cellular automata (CA) with the adaptive potential of Physarum slime mold. Namely, we propose a CA model and multi-agent approach to imitate the behavior of the plasmodium. We then test the efficacy of the proposed model on graph problems such as the maze problem or the traveling salesman problem (TSP). Finally, the virtual Physarum model is evaluated on a data set for pattern recognition purposes and achieves to form very effectively the letters of the alphabet, especially when compared with real experiments performed to prove the efficacy of the proposed model. Furthermore, to exploit the CA's inherent parallelism and make the model's responses faster, both GPU and hardware implementations are proposed and compared. As a result, an accelerated virtual lab is developed which uses a multi-agent CA model to describe the behavior of plasmodium and can be used as an intelligent, autonomous, self-adaptive system in various heterogeneous and unknown environments spanning from different types of graph problems up to real life-time applications.

INDEX TERMS Cellular automata, GPGPU, multi-agent models, parallel computing, models, Physarum Polycephalum, virtual lab.

I. INTRODUCTION

Physarum polycephalum is a slime mold forming a large single cell that can be found in cool, moist areas such as decaying leaves and logs. It has many phases in its life cycle, but the most interesting one for computation-oriented research is its vegetative stage called plasmodium. Plasmodium has no central control or no brain to deliver instructions globally to every part of its mass. It is comprised of many tube-like structures that communicate locally and let electric signals move from one place to another. Macroscopically, it looks like a dendritic network of connected tubes which expand in the given space area, trying to reach food in order to survive. Those expanding tubes are called pseudopodia. It is found that it has the ability to connect two food sources (FSs) existing in two different points, in a minimum distance. This is rational in the sense that the organism needs to create the minimum path between the FSs to minimize energy spent on transporting nutrients between two parts of its body. This Physarum's characteristic was an inspiration to many scientists for solving complex mathematical problems. More specifically, Nakagaki *et al.* [1] showed that this simple organism has the ability to find minimum-length solution between two points in a labyrinth. Adamatzky [2] proves that slime mold has the ability to find the path between two sites in one pass, assisted by the gradient of chemo-attractants. The research on slime mold was not limited only in maze problems. Subsequent research has enabled slime mold computational abilities to apply in various problems like spatial representations of various graph problems [3]–[8], robotic control [9], [10], biological electronics [11]–[13], and more [14]. The experiments of the living Physarum may last several hours or

The associate editor coordinating the review of this manuscript and approving it for publication was Shubhajit Roy Chowdhury.

days, which can be considered as a major drawback in its computational abilities. The key to unlock and further exploit those abilities is the modeling approximation of the organism as precise and as fast as possible. Until now, there is a variety of modeling approaches since no single model that can still describe exactly the whole behavior of Physarum, even considering only the plasmodium stage. Current attempts at modeling Physarum's behavior try to simplify this huge task by compartmentalizing the different behaviors of the organism in different situations. For example, there are publications trying to model the mechanisms of growth [15], the movement [16], the internal oscillations [17], [18] or the network adaptation [19]. So, it is obvious that the choice of the right modeling tool is a substantial step toward the reproduction of the biological substrate's behavior as close to reality and as fast as possible.

Having in mind principles and characteristics prominent to natural and physical phenomena, like randomness, heterogeneity and local interactions, CA sound a very promising computational tool to deal with. In an abstract but adequate definition of CA, these are models of physical systems, where space and time are discrete and interactions are local. CA combine the use of local memory (CA cell state) and processing unit (CA local rule) in the same place, i.e. a CA cell. They can capture the essential features of systems, where global behavior comes out as the collective effect of simple components, which interact locally. In addition, they can adequately handle complex boundary and initial conditions, inhomogeneities and anisotropies. These characteristics are very convenient for modeling physical systems and particularly to simulate the behavior and dynamics of a biological organism such as *Physarum Polycephalum*. CA have been successfully used in a vast range of research fields such as traffic control [20]–[22], fire spreading [23], [24], molecular dynamics [25], [26], logic gates design [27], pedestrian dynamics [28], [29], crowd evacuation [30]–[33], biological systems [34], [35], or even Physarum modeling [36], [37].

On the other hand, Agent-Based Models (ABMs) are focused on the behavior of its individuals in the system. The base component of such a model is the 'agent' which represents an entity and tries to mimic its characteristics when interacting with other agents or with its environment. The global behavior arises from the interaction between simple components and in a sense it sounds similar to the CA cell state and the rule that controls its behavior. The advantage that an ABM offers is that it offers a real-time observation of each entity in the system instead of just a general view of the results at the end. In other words, the modeling goal is not the high-level behavior but the exact individual's behavior over time. In this way, it is easier to capture the interactions, the randomness and the heterogeneity of the system mentioned before. Agent based models (ABMs) are used very often in order to describe system flows [38], energy systems [39], biological systems [40]–[42], and more specifically Physarum behavior [43]–[46].

In this paper, we combine the Physarum computing abilities, the CA modeling tool and the agent based modeling approach to design a hybrid CA Agent Based Model, which accurately approximates the vegetative behavior of Physarum Polycephalum. Firstly, we apply our model to the problem of the maze solving, i.e. the ability of the plasmodium to find the minimum path between two sites as a proof of model's efficacy. The biological parts of slime mould are described by CA agents which interact with each other under certain CA rules. The results show that the proposed model develops successfully the minimum-distance solutions by creating solid tubes of sequential agents between two sites in a maze. Furthermore, we stress further the proposed model to deliver solutions to the well known Traveling Salesman Problem (TSP). TSP is represented in the CA lattice with many food spots (FSs) as the imaginary position of the nodes, as well as many agents which represent the different parts of plasmodium. Those CA agents are moving according to the CA rules and they succeed to provide solutions to the TSP problem very effectively by creating solid tubes between the virtual cities. The results show once again model's ability to provide solutions to TSP in a small number of cities. Finally, the CA agent model is tested in a different challenge, namely is used to evaluate its ability to form other graph shapes such as the 26 letters of the alphabet. ÎIJoreover, the CA and ABM (agent based model) inherent parallelism is exploited to make model's responses faster. More specifically, both parallel GPU and FPGA implementations of the CA agent model are thoroughly presented and compared in details. As a result, an accelerated virtual lab is proposed which facilitates a new CA ABM model to describe the behavior of plasmodium and can be used as an intelligent, autonomous, self-adaptive system in heterogeneous and unknown environments such as the different versions of TSP or maze-solving systems.

II. THE PROPOSED CA ABM FOR PHYSARUM'S BEHAVIOR

In this section, the CA-ABM model for describing and modeling the behavior of *Physarum Polycephalum* is presented. In particular, each one of these particles is represented by a CA agent. All these CA agents and their behavior in space and time result to the general behavior of the plasmodium. The area where the experiment takes place is also divided into a matrix of squares with identical areas and each square of the surface is represented by a CA cell. In case of the first model's test, namely the labyrinth, two virtual FSs are placed in two different sites of the maze and plasmodium is placed on top of both. In the biological experiment, the plasmodium will start traveling in every direction inside the maze and after some time steps it will potentially cover almost every part of the maze. Then it will start to shrink and will create the main protoplasmic tube between the two FSs. This is the starting point of our algorithm, when the cells/agents of the plasmodium stopped the expansion and are ready to shrink and create the tube. Those agents are placed inside the area in random sites of the CA lattice, where there is free space and not a wall. In the case of proximity graphs, many FSs are placed in different sites of the area given. The state of the i, j cell at time t, defined as $C^t_{i,j}$ is equal to:

$$C^t_{i,j} = \{Flag_{i,j}, Food^t_{i,j}, Ask^t_{i,j}, Rand^t_{i,j}, Agent^t_{i,j}\} \qquad (1)$$

where $Flag_{i,j}$ is a variable that can acquire four (4) different values and indicates the type of the area represented by the corresponding i, j cell. The possible values of $Flag$ are the following ones:

- $Flag$="00" is considered as a free area.
- $Flag$="01" is considered as the area of initial placing a FS.
- $Flag$="10" is considered as the area of initial placing the agents which represent particles of the plasmodium.
- $Flag$="11" is considered as an area which represents the walls of the maze (agents cannot move to such a cell).

Moreover, $Food_{i,j}$ represents the concentration of chemo-attractants at time t in the area corresponding to the i, j cell. The expansion of chemo-attractants are considered uniform, so the diffusion equation translated to the CA language is given by equation 2:

$$\begin{aligned} Food^{t+1}_{i,j} = \Big\{ & Food^t_{i,j} + fp1\Big[\Big(Food^t_{i-1,j} - fp3 \times Food^t_{i,j}\Big) \\ & + \Big(Food^t_{i+1,j} - fp3 \times Food^t_{i,j}\Big) \\ & + \Big(Food^t_{i,j-1} - fp3 \times Food^t_{i,j}\Big) \\ & + \Big(Food^t_{i,j+1} - fp3 \times Food^t_{i,j}\Big)\Big] \\ & + fp2\Big[\Big(Food^t_{i-1,j-1} - fp3 \times Food^t_{i,j}\Big) \\ & + \Big(Food^t_{i+1,j-1} - fp3 \times Food^t_{i,j}\Big) \\ & + \Big(Food^t_{i-1,j+1} - fp3 \times Food^t_{i,j}\Big) \\ & + \Big(Food^t_{i+1,j+1} - fp3 \times Food^t_{i,j}\Big)\Big]\Big\} \qquad (2) \end{aligned}$$

The numerical values of the fitting parameters of equation 2 indicating the influence of the adjacent neighboring cells, are heuristically given as $fp1 = 0.05$, $fp2 = 0$ and $fp3 = 1$, respectively.

Furthermore, $Ask_{i,j}$ is a variable that can take eight different values from 1 to 8 while checking if an agent exists in the corresponding CA cell. If there is, it searches its adjacent eight (8) neighbors (north-west, north, north-east, west, east, south-west, south, south-east — forming a classical Moore neighborhood, and finds which one of them has the greater $Food$ value. The value of $Ask_{i,j}$ is depended on which of the neighbors in this Moore neighborhood has the greater $Food$ value. For example, if the north-west neighbor has the greater value, $Ask_{i,j}$ takes the value 1. If the north neighbor has the greater value, $Ask_{i,j}$ will take value 2. If the north neighbor has the greater value, $Ask_{i,j}$ will take value 3 and so on. This procedure for all possible cases is presented graphically in Fig. 1.

$Rand^t_{i,j}$ is a variable that delivers a new pseudo-random number in each time step. This variable was chosen in order

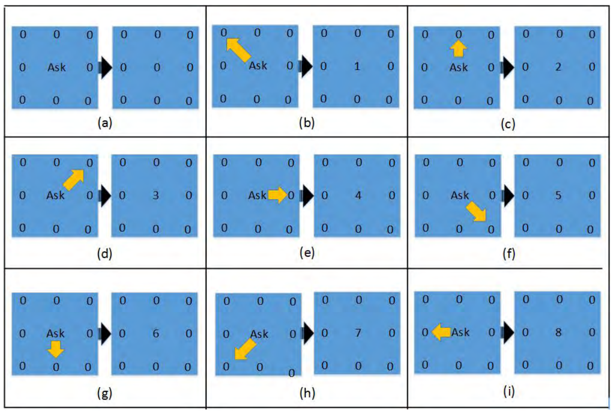

FIGURE 1. $Ask^t_{i,j}$ variable numerical interpretation and visualization.

to be able to decide which agent is going to occupy a specific place when there is a conflict with other agents. So, every CA cell of the lattice holds a pseudo-random number which is updated and renewed in each time step.

Finally, $Agent^t_{i,j}$ is a variable that declares the existence or not of an agent. It is the variable that defines if an agent is going to leave an occupied CA cell or not. It also defines if an agent is going to take an empty cell or not. This procedure is performed as follows: If a specific cell, with an agent inside it ($Agent^t_{i,j}$=1), has a value of variable $Ask^{t-1}_{i,j}$=3, which means that asks for moving to north-east cell, and there is no other agent that requests to go in the same cell, then $Agent^t_{i,j}$ takes the value 0 while the north-east cell takes value $Agent^t_{i-1,j+1}$=1 (Fig. 2 a). If there is a conflict with another (Fig. 2b) or many other agents (Fig. 2c) who ask to move on this specific cell, then it checks which one of them has the maximum $Rand^t_{i,j}$ value. If $Agent^t_{i,j}$ has the maximum $Rand^t_{i,j}$ value then $Agent^{t+1}_{i,j}$ takes value 0 and the $Agent^{t+1}_{i-1,j+1}$ takes value 1. In last case, that it does not have the maximum value, then it keeps the value $Agent^{t+1}_{i,j}$=1.

III. GPU DESIGN OF THE CA-ABMODEL

In this section the GPU implementation of the proposed model are discussed analytically. The basic idea is to exploit the inherent parallelism of CA as modeling tool and CA ABM as a parallel modeling technique. GPU using CUDA is a very powerful tool, that helps the programmer to design and implement a parallel model very easily. It is well known that GPUs have been widely used from researchers for accelerated modeling techniques [47]–[51]. In this implementation, the initial CA data are stored to the global memory of the device. It would be probably more efficient to use the shared memory, which is faster, but for simplicity purposes this programming approach will be discussed in one of our future works. The main steps of our algorithm are:

1) Split the CA sub-states into different blocks of threads. First, a 2-dimensional block for the definition of the modeling morphology and environment is created while, a 2-dimensional block for the calculation of the chemo-attractants' diffusion, a 2-dimensional block

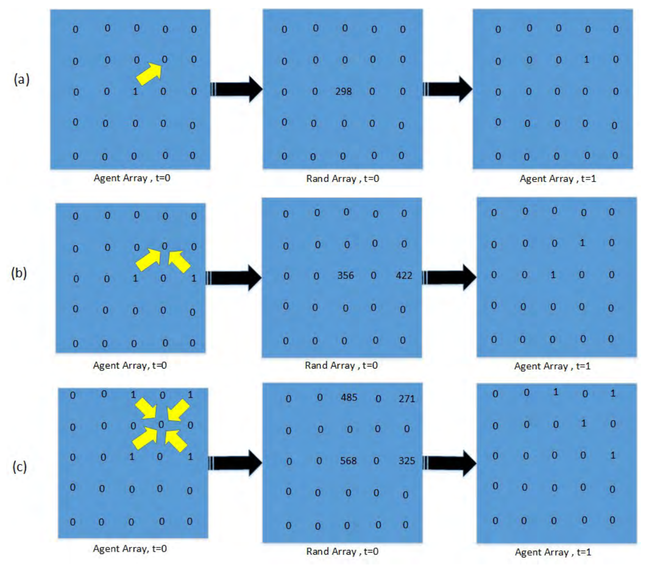

FIGURE 2. Graphical examples of $Agent^t_{i,j}$ procedure. (a) Only one (1) agent seeks to move to north-east cell. (b) Two (2) agents compete for the same cell. (c) Four (4) agents compete for the same cell.

which holds the $Ask^t_{i,j}$ value for every agent at each time step, a 2-dimensional block to update its cell with the $Rand^t_{i,j}$ values and, finally, a 2-dimensional block, which calculates the movement of each agent, are also created.

2) Then, assign a kernel to process and make the necessary calculations for every block mentioned before. More specifically, a kernel to hold and update the $Flagi, j^t$ of the CA cell, a kernel for the computation of the discrete diffusion equation of the chemo-attractants, a kernel for the calculation of the $Ask^t_{i,j}$ variable, a kernel responsible for the updating of the random values of $Rand^t_{i,j}$ variable and, finally, a kernel that calculates the movement or not of each CA agent are assigned respectively.
3) Initialize the current state for all the kernels through a CPU-GPU memory copy operation, from host-CPU memory to global memory of the GPU device.
4) Run the appropriate kernel and make the calculations by using the information of the current block of the CA variable.
5) At the end of each time step, assign other kernels to make the device to device memory copy operation. This procedure updates the $CA_{current}$ block with the CA_{next} block.
6) Finally, complete the simulation, while the final block of the CA is being retrieved from the global memory of the device to present the results.

The processing procedure of the host and the device in order to complete a full cycle of all CA variables is presented in Fig. 3.

IV. FPGA DESIGN OF THE CA-ABMODEL

In contrast to the serial computers, the model hardware implementation has the motive of parallel processing, which is a

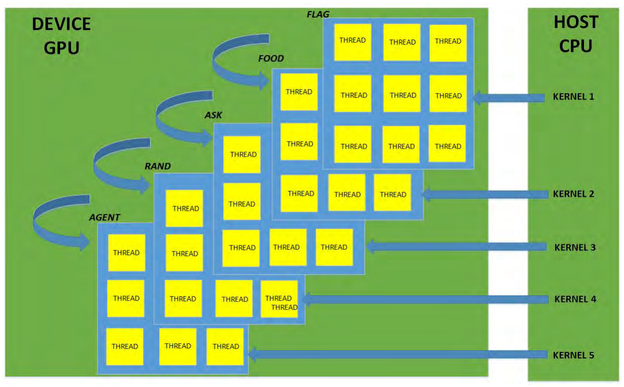

FIGURE 3. In each time step every kernel receives a signal from the host to start the procedure of its block. Every block handles a different CA state. A whole cycle of the CA rule is finished when the fifth kernel finishes the calculation of the $Agent^t_{i,j}$ block.

TABLE 1. The flow summary for the basic ca cell produced by Quartus software.

Quartus II 32-bit Version	12.1 Web Edition
Top-level Entity Name	PhysarumCell
Total Logic Elements	1721
Total Registers	17

TABLE 2. The max frequency of the implementation at 85 °*C* and 0 °*C* as produced by Quartus software.

Temperature	Fmax	Restricted Fmax	Clock Name
85 °C	14.85Mhz	14.85Mhz	clk
0 °C	16.21Mhz	16.21Mhz	clk

native characteristic of CA and agent based models as well as the biological organism we test. This can obviously lead to further acceleration of the proposed model. To this end we propose the hardware implementation of the proposed model to an FPGA device. More specifically, the VHDL hardware description language was used, while the design and the processing of the circuit have been made using Quartus software and the simulations were done by using Modelsim of the ALTERA company.

The signals used in this circuit as inputs or outputs are the same as those in the GPU implementation. Each main CA cell, or else an FPGA basic block uses approximately 1721 logic elements. The logic elements, the pins, and the registers used for the basic CA block is presented in Table 1.

The frequencies achieved at 85 °C and 0 °C corresponding temperatures, are presented in Table 2. As it is obvious, the max frequency at 85 °C was 14.85 Mhz, while at 0 °C was 16.21 Mhz. So, the frequency speed achieved is approximately 15 Mhz or, in other words, every time step of our model needs approximately 66 ns. The Power Analyzer tool of Quartus showed 87.3mW power consumption.

The basic structure of this implementation is called "`PhysarumCell`" block (Fig. 4). Every "`PhysarumCell`" block is connected with its 24 closest neighbors and is respon-

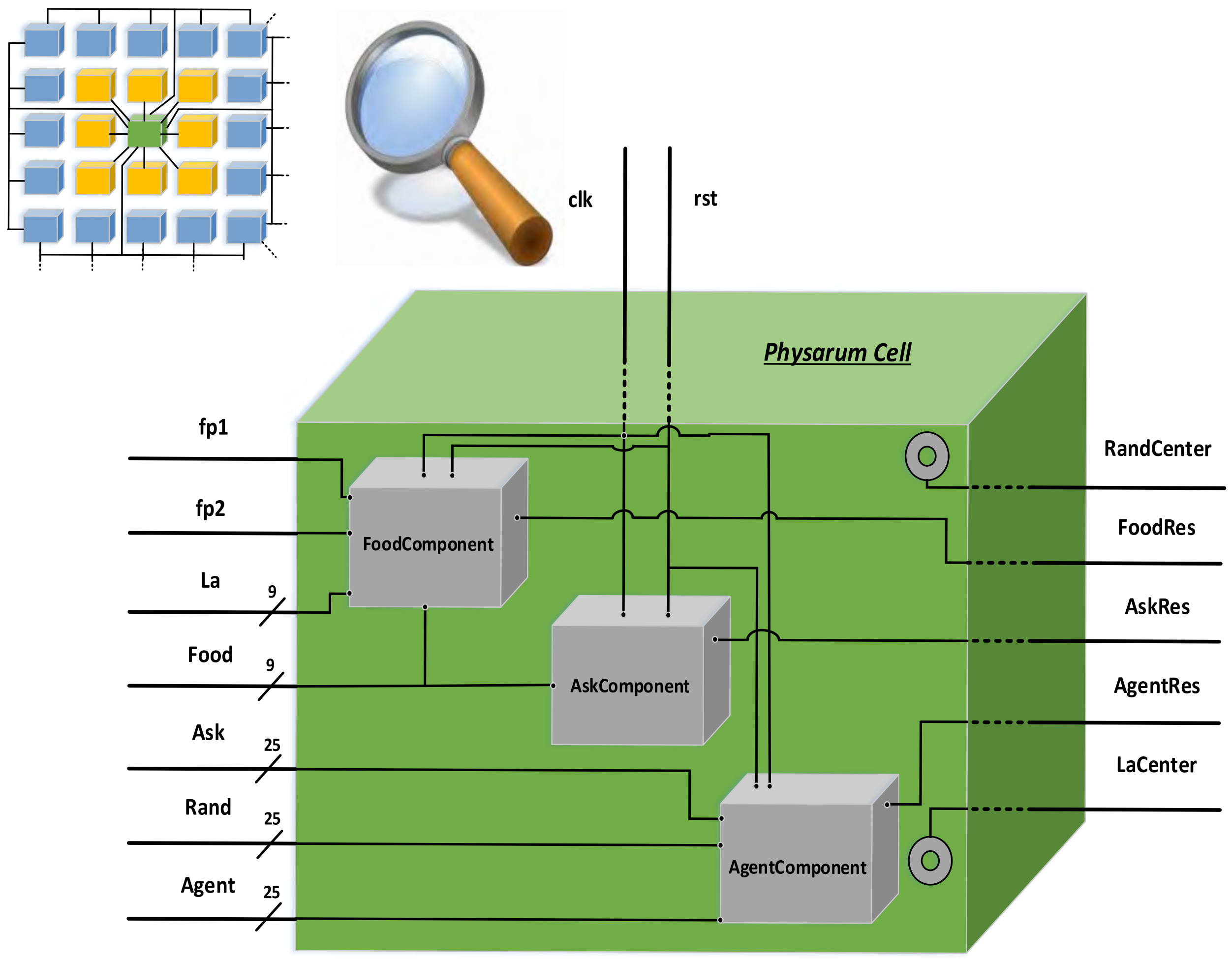

FIGURE 4. `Physarum Cell` **is the basic CA cell in the CA lattice and the basic block in the FPGA.** `FoodComponent` **is the component responsible for the calculation of the chemo-attractants' diffusion.** `AskComponent` **is the component responsible for the calculation of $Ask^t_{i,j}$ variable, while** `AgentComponent` **is the component responsible for the calculation of the agents' movement.** `LaCenter` **and** `RandCenter` **are internal cell's states which refer to the type of the cell's topology and random number respectively.**

sible for handling the signals that try to simulate the behavior of the plasmodium. Figure 4 shows that this block has 97 inputs and 5 outputs. The inputs are:

- 2 circuit signals, 1 bit each, the clock and the reset signal.
- 9 signals, 2 bit each, which represent the morphology of the neighbors' and the central cell's environment.
- 9 signals, which are integers ranging from −10,000 to 10,000 and represent the concentration of the chemo-attractants of the neighbors and the previous concentration in the central cell.
- 2 signals which are integers ranging from 0 to 100 and represent the parameters of the chemo-attractants' diffusion equation.
- 25 signals, which are integers ranging from 0 to 8 that hold the neighbors' values and the central value of the $Ask^t_{i,j}$ variable.
- 25 signals, which are 1 bit each and represent the existence or not of an agent in the 24 closest neighbors and that one in the central cell.
- 25 signals, which are integers ranging from −100 to 100 that represent the random values of the neighbors. These values are used in the case of a conflict during the agents' movement.

The output signals are:

- 1 signal, which is an integer ranging from −10,000 to 10,000 and represents the concentration of the chemo-attractants in the current cell.
- 1 signal, which is an integer ranging from 0 to 8 and holds the value of $Ask^t_{i,j}$ of the current cell.
- 1 signal, which is 1 bit signal, and represents the existence or not of an agent in the CA cell.
- 1 signal, which is an integer ranging from −100 to 100 that represents the random value that the current cell holds.
- 1 signal, which is a 2-bit signal and represents the type of the current cell.

Every "`PhysarumCell`" block consists of three components, the *FoodComponent*, the `AskComponent`, and the `AgentComponent`, respectively as shown in Fig. 4. Each one of the aforementioned component is responsible for a specific task, i.e to calculate the diffusion of the chemo-attractants or the movement of the agents. More specifically, the `FoodComponent` is a component responsible for the calculation of the chemo-attractants diffusion. As an input it takes the type of the CA cells of its 9 neighbors with their type, the 9 chemo-attractants concentrations of its neighbors with their concentration in the previous time step, the 2 parameters of chemo-attractants diffusion equation, and 2 circuit signals, namely the `clk` and the `rst`. Afterwards it calculates the result of the diffusion equation and updates its current value with the new one. The `AskComponent` is a component responsible for the calculation of the current agents direction. In particular, it uses the chemo-attractants' concentration of its 9 neighbors and two signals for synchronization, i.e. the `clk` and the `rst`. It checks from which direction comes the greater amount of chemo-attractants and produces in the exit the appropriate integer which declares the desired direction. Those possible numbers in the exit can be summarized as follows:

- "0", if there are no chemo-attractants in its neighbors.
- "1", if the chemo-attractants of the north-west neighbor have the greatest concentration.
- "2", if the chemo-attractants of the north neighbor have the greatest concentration.
- "3", if the chemo-attractants of the north-east neighbor have the greatest concentration.
- "4", if the chemo-attractants of the east neighbor have the greatest concentration.
- "5", if the chemo-attractants of the south-east neighbor have the greatest concentration.
- "6", if the chemo-attractants of the south neighbor have the greatest concentration.
- "7", if the chemo-attractants of the south-west neighbor have the greatest concentration.
- "8", if the chemo-attractants of the west neighbor have the greatest concentration.

Finally, the `AgentComponent` is the component responsible for the calculation of the agents' movement. As an input it takes the circuit signals (`clk`, `rst`), the *Ask* and the *Agent* value of the 25 closest neighbors. This happens because the central cell must have the information of the position and the selection of direction of all the neighbors around it. Then it considers all these as described in II and the output is a 1-bit signal that takes value 1 if an agent comes or stays to the cell and 0 if an agent is leaving the cell or the cell stays empty.

V. MAZE IMPLEMENTATION RESULTS

As proposed in the introduction, firstly we applied our model in order to solve the maze presented by Nakagaki et al. [1]. This is why we focus on the shrinking behavior of the plasmodium's body after its expansion. We have used probability rules to locate the agents in their initial positions to simulate the actual distribution of Physarum's body in the experiments and we have also used probability in their movement, because in the case of conflict with many agents', the agent with the greater $Random^t_{i,j}$ value will eventually move. Usually, the vast majority of CA ABMs are stochastic, which means that two simulations will not normally reproduce exactly the same data. However, when repeating many runs, the qualitative behavior should result with the same qualitative characteristics in most of them. So, in order to achieve the reproducibility of the model, we tested our model for three different initial conditions. At the beginning we tested a labyrinth, in which the initial agents had 30% probability to exist or not in a specific cell with 0 as value for $Flag_{i,j}$. Then 10 experiments have been made in order to ensure that reproducibility occurs. Afterwards, the same 10 tests have been made with 50% probability and finally another 10 tests run with 70% probability.

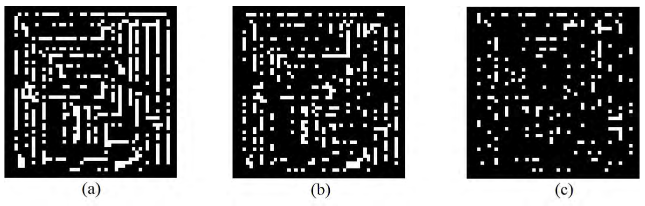

FIGURE 5. The initial values of the $Agent^t_{i,j}$. (a) The agents have 70% probability to take an initial position at a free space of the labyrinth. (b) The agents have 50% probability to take an initial position. (c) The agents have 30% probability to take an initial position.

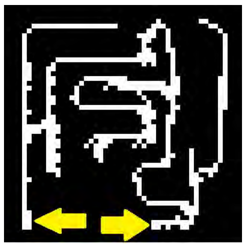

FIGURE 6. The yellow arrows present the sites of the maze from where the chemo-attractants start to expand.

In Fig. 5, the initial state of the $Agent^{t=0}_{i,j}$ is presented with probability 70% (5*a*), 50% (5*b*) and finally 30% (5*c*). The initial food is placed in two sites of the maze as shown in Fig. 6. These spots are also shown in Fig. 6, to the edges of the yellow arrows. Then, the model begins to run until no agents move from their position any longer.

The first 10 results with 70% probability for the initial agents are shown in Fig. 7. The model run approximately for 5,000 steps for every one of the 10 tests. The average finishing time of the experiment is approximately 4.503s.

The model finds the minimum distance between the two FSs in 9 to 10 experiments. If we follow the only closed path from the first source of nutrients, we are going to reach other source of nutrients along a minimum distance path. This is achieved in the experiments illustrated

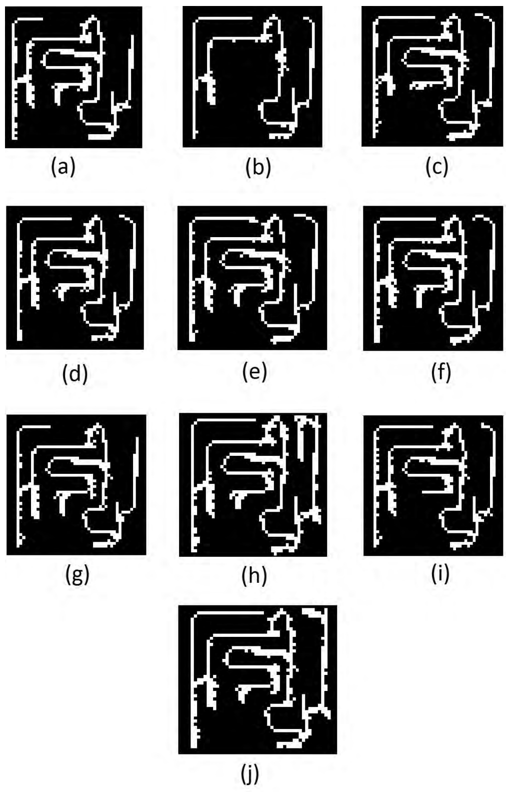

FIGURE 7. **(a)-(j) The 10 different tests for 70% probability for the initial state.**

in Fig. 7($a-d$ and $f-j$). Only the experiment in Fig. 7(e) fails to find the minimum distance, because the path is open at a specific point.

The other 10 results with 50% probability for the initial agents are shown in Fig. 8. The model runs approximately for $4,500$ steps for every one of the 10 tests. The average finishing time of the experiment is approximately 3.998s. The model finds the minimum distance solution in the maze in 6 to 10 experiments, which can be considered as a success having in mind the original distribution percentage. The experiments which succeeded are presented in Fig. 8(d, e, f, h, i, j). Those who have failed because there is an open path between the two FSs are presented in Fig. 8(a, b, c, g), respectively.

The final 10 results with 30% probability for the initial agents are shown in Fig. 9. The model run approximately for $4,000$ steps for every one of the 10 tests. The average finishing time of the experiment is approximately 3.613s. These tests failed because just one succeeded to find the minimum path between the two FSs. The experiments which failed are presented in Fig. 9($a-f$ and $h-j$) and the successful experiment is illustrated in Fig. 8(g).

As it was mentioned in Section IV, a CA cell in our FPGA implementation uses approximately 1721 logic elements. Therefore, in order to create a more practical and applicable simulation in a real FPGA device, with almost limited computational resources, we have created a 23×23 grid with a total of 910,409 logic elements for the

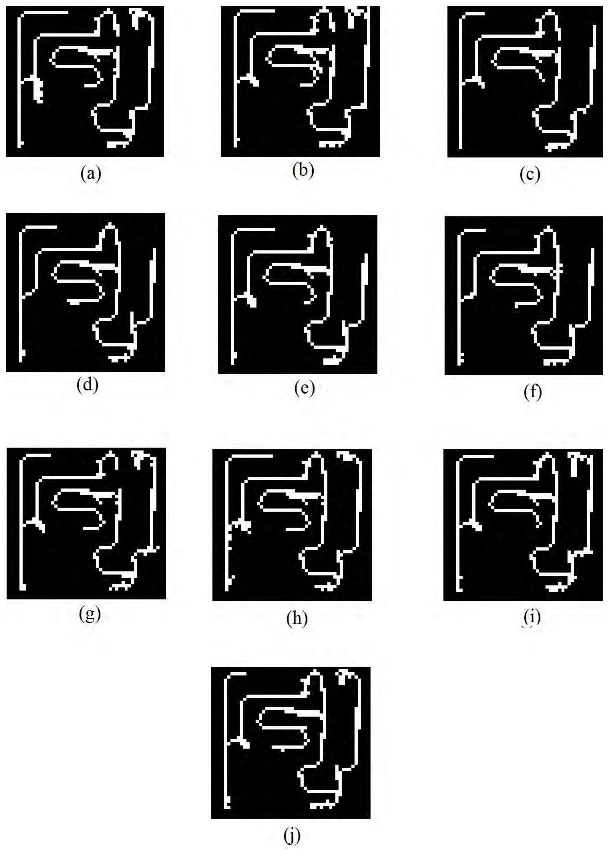

FIGURE 8. **(a)-(j) The 10 different tests for 50% probability for the initial state.**

whole grid. For the time being, there are high-end FPGAs in the market that come up with 952,000 logic elements, i.e. 28nm Stratix V.

The same initial conditions like the ones implemented in the software tests, were used in the FPGA implementation. The results produced by ModelSim were the same with those produced by CUDA. More specifically, with 70% initial agent probability the model, also, achieves to find the minimun path in $9/10$ experiments. With 50% initial agent probability it achieves $6/10$ successful experiments and with 30% initial agent probability, achieves $1/10$ successful experiments. But as we mentioned in the previous sector, our hardware design achieves 18.6 Mhz frequency. This means that the model needs approximately 53.7 ns.

So, for 5,000 time steps FPGA needs approximately 268.5 μs, for 4,500 needs 241.6 μs and for 4,000 needs 214.8 μs. In this way, it achieves a speed-up of approximately six orders of magnitude against the CUDA implementation.

VI. CREATING PROXIMITY GRAPHS

The next step is to test the above model of Physarum in more classical graphs. Toussaint [52] used three types of graphs, the Minimum Spanning Tree (MST), the Relative Neighborhood Graph (RNG) and the Delaunay Triangulation (DT) and proposed an hierarchy of them. The MST [53] is a connected acyclic graph which has the minimum possible

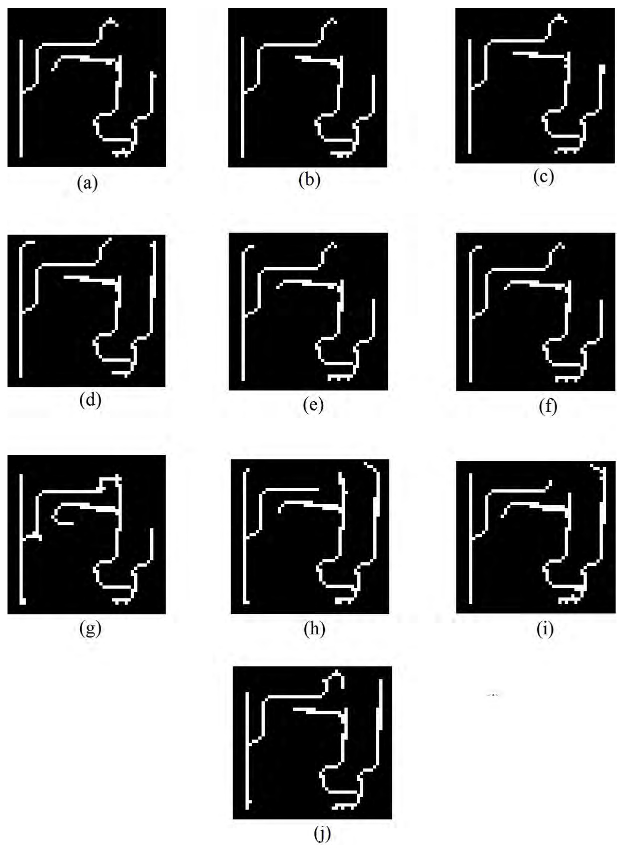

FIGURE 9. **(a)-(j) The 10 different tests for 30% probability for the initial state.**

sum of edges' lengths. The RNG is a graph that uses relative neighbors to form its edges. Given a set of points $P = \{p_1, p_2, p_3, ..., p_n\}$ a point p_i is relatively close to another p_j if $d(p_i, p_j) \leq max[d(p_i, p_k), d(p_j, p_k)]$ for all $k = 1, 2..., n$, $(k \neq i, j)$ where $d(p_i, p_j)$ is the distance between points p_i and p_j. DT [54] is a graph subdividing the space onto triangles with vertices in V, edges in E, where the circumcircle of any triangle contains no points of V other that its vertices. Toussaint showed that $MST \subseteq RNG \subseteq DT$. In this hierarchy, the Gabriel Graph (GG) [55] was inserted between RNG and DT, in which two points p_i, p_j of the graph form an edge if the circle having this edge as diameter is empty from any other points of the graph. In [56], Adamatzky showed that the plasmodium of *Physarum Polycephalum* has the ability to mimic the Toussaint hierarchy. He demonstrated that the initial conditions of the experiment were crucial for the results and depending on them plasmodium has the ability to form different types of graphs. For example, if nodes (nutrients) are placed in a petri dish and the plasmodium is placed in one of them to start its foraging behavior then after some time steps it will probably create an RNG graph. If the plasmodium is placed in every node at the start of the experiment, then it is much more likely to form graphs closer to GGs. In our model we make use of the second case. We assume that the plasmodium is placed in every food source which represents a vertex in the graph. At the first stage the plasmodium starts

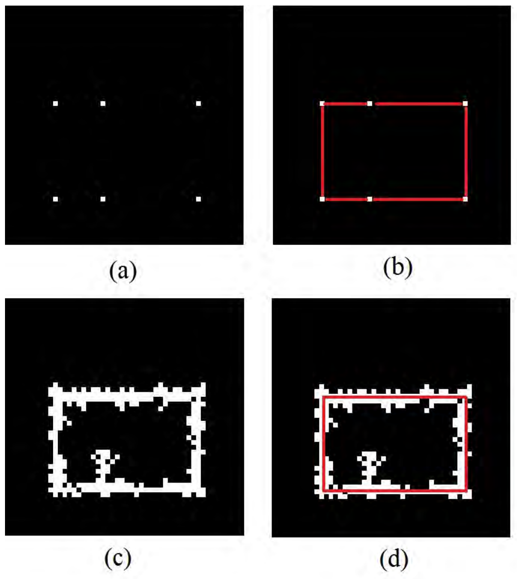

FIGURE 10. **Test of the proposed model in 6 cities. (a) The initial position of the cities. (b) The ideal solution. (c) The results of the experiment. (d) The closed route that determines the path between the cities.**

to expand in every direction covering all the area around a FS. At the next stage, which is the starting point of our model, it is affected by the chemo-attractants of other FSs, it shrinks and starts creating protoplasmic tubes between them.

In this implementation, we stress our model to see how well it would behave in a simple TSP problem with a few and many nodes/cities that are placed accordingly to form either relative neighborhood or Gabriel graphs. We decided to use 70% probability for the agents' initial state, because as presented in Sect. V it produced better results. The following experiments managed to display their results in the majority (>60%) of the repeated tests.

First, we tested six nodes to check if a rectangle is going to be created. The results are presented in Fig. 10. In Fig. 10(*a*) the initial position of the cities is presented and in Fig. 10(*b*) the ideal solution is presented. In Figs. 10(*c*, *d*) the result of the experiment and the painted closed path are depicted. It is obvious that in this simple problem, the model manages to reproduce the same result as the ideal one.

In order to compare the results of our model in similar simple problems we created an experiment with 5 cities that are supposed to form an easy shape as a solution. The experiment is presented in Fig. 11. In Fig. 11(*a*) it is presented the initial position of the cities, in Fig. 11(*b*) the ideal solution to this problem, in Fig. 11(*c*) the result of the experiment and in Fig. 11(*d*) the path that the agents form. The experimental results seem to connect all the cities in one pass but the distance is not the minimum. It has a small diversion from the ideal path. This proves that the success of an experiment depends heavily on the initial position of the cities, the initial position of the agents and the randomness in their movement.

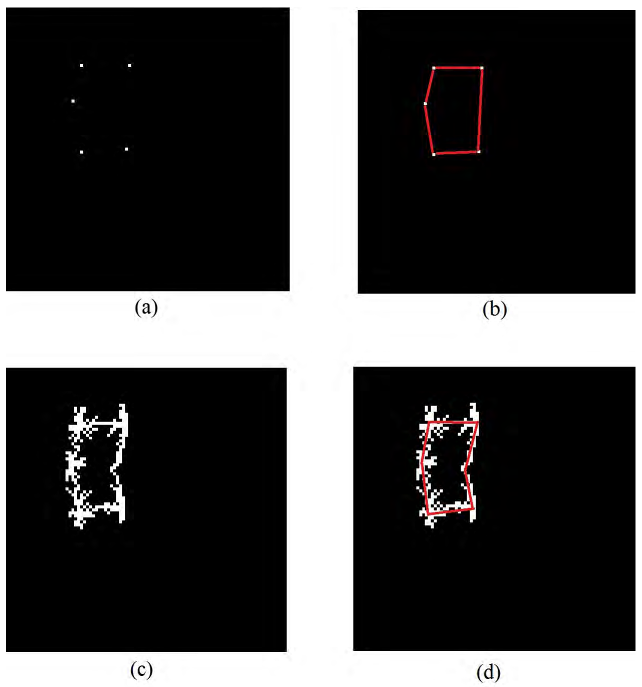

FIGURE 11. Test of the proposed model in 6 cities. (a) The initial position of the cities. (b) The ideal solution. (c) The results of the experiment. (d) The closed route that defines the path between the cities.

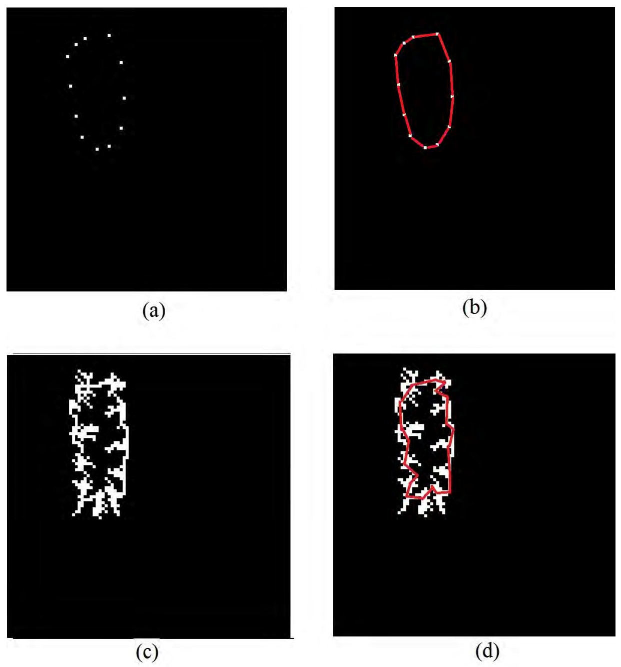

FIGURE 12. Test of the proposed model in 6 cities. (a) The initial position of the cities. (b) The ideal path. (c) The experimental results. (d) The path that the experiment produces.

So, when we have to deal with more complicated, or different problems, the results shall be also different.

The next more advanced experiment involved 12 cities that will produce a circular shape route. For one more time, we put agents in an area that covers all the cities with 70% probability as before. The experiment and the results are presented in Fig. 12. In Fig. 12(a) the initial cities' configuration is shown. In Fig. 12(b) the ideal path produced by Concorde software [57] is depicted. Then, in Fig. 12($c - d$) the result of the experiment and the closed path (in order to be more understandable) are graphically given. Once again, we can check that the execution of the proposed model manages to find a path that connects all the cities at least once, but there is also a small divergence between the experimental and the ideal result in terms of distance.

A final example is the one given in Fig. 13. As before, the initial cities' position is presented in Fig. 13(a), while the ideal path produced by Concorde software is shown in Fig. 13(b). Then, in Fig. 13($c - d$) the results of the experiment and the closed path are presented. The model finds the path that connects all the cities at least once, but there is also a greater divergence between the experimental and the ideal result in terms of distance, greater than before. The same philosophy and results are presented in a more complicated, in terms of shape, problem in Fig. 14.

Based on the provided results, we can say that the proposed model can be applied on a travelling salesman problem if the cities/nodes/FSs are placed in such positions that form relative neighborhood or Gabriel graphs. However, in other graphs that do not meet those criteria, our algorithm will probably fail in most cases. Those simulations were tested both in serial and parallel processing units. The processing time ranges between $1 - 2$ seconds running on one core of a 3^{rd} generation Intel CPU while the FPGA needs approximately $80.5\mu s - 107.4\mu s$.

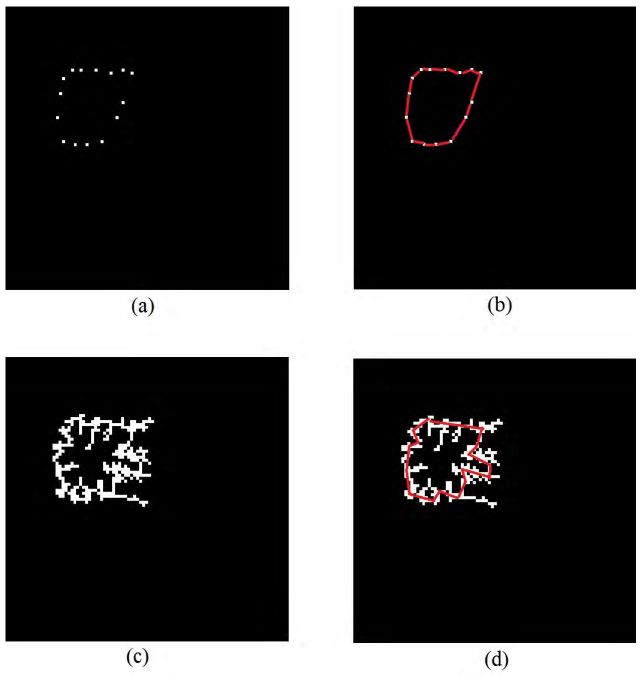

FIGURE 13. Test of the proposed model in 12 cities. (a) The initial position of the cities. (b) The ideal path. (c) The experimental results. (d) The path that the experiment produces.

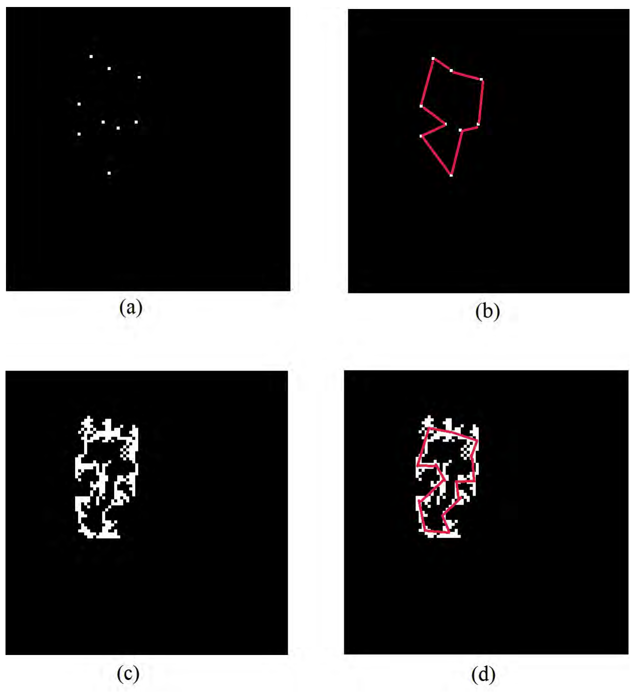
FIGURE 14. Test of the proposed model in 15 cities. (a) The initial position of the cities. (b) The ideal path. (c) The experimental results. (d) The path that the experiment produces.

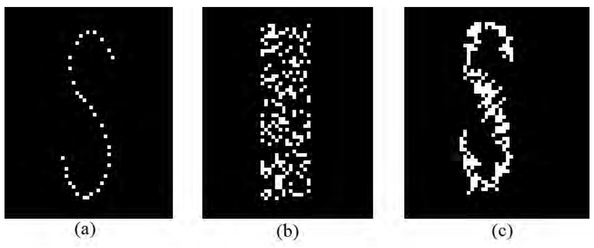
FIGURE 15. An example of forming the letter 'S'. (a) The initial position of the nutrients. (b) The agents are placed with 30% probability within the area that the nutrients are forming. (c) The final outcome; the agents form the letter 'S'.

VII. PHYSARUM LEARNS THE ALPHABET

In the previous section, it was shown that the CA agent model of Physarum has the ability to resemble proximity graphs by constructing agent-veins to connect a number of stimuli. The nodes of these networks are represented by food spots and the edges are represented by the agent-chains forming virtual tubes. In this section, our model's behavior is tested, in the case that the virtual plasmodium agents are inoculated in a lattice to interact within a data-set, forming the letters of the English alphabet.

At the beginning, the stimuli are placed properly in a lattice of 10, 000 cells without any walls or repellents so as to form each letter. The number of nutrients ranges between 20 and 40 and is proportional to the complexity of each letter. At the next step the minimum and maximum (x, y) coordinates of the stimuli's covered area are calculated. Then, the initial CA agents are placed within this area randomly. The initial probability parameter was chosen to 30% for every cell of the lattice. It should be noted that in this case, we didn't want to increase the population of the agents because an overcrowded space results in removal of the internal space and transition from an α-shape to a solid Concave Hull as happens for example during the formation of letter 'A'. A smaller amount of agents would lead to an increasing number of discontinuities between the nutrients.

In Fig. 15 the whole procedure is shown while forming the letter 'S'. In Fig. 15(a) the nutrients are placed in the lattice in order to create the letter 'S'. In Fig. 15(b) the CA agents are placed inside the area that the Food spots are forming with 30% probability. Then, the model is running for 1, 500 time steps and the final result is presented in Fig. 15(c). However, there are some small defects and the line is not purely clear. but it is obvious that the CA agents managed to approximate very closely the desired outcome.

In Fig. 16 all the produced letters from 'A' to 'Z' are presented. At first sight, it is clear that the proposed model achieves to form all the shapes of the letters, although, as expected, some letters are formed better than others. For example, the letters 'Q', 'O', 'S', or 'U' are very clear. On the other hand, letters such as 'I', 'J', 'Y' or 'T' are also formulated but with less clear way when compared with the others. Those letters are closer to minimum spanning trees and could be formed more precisely if the plasmodium was placed in only one FS as suggested in [56]. So, even if the CA agent model seems to be impressionable to the initial conditions and the characteristics of each letter, it manages to approximate very closely the desirable result.

Furthermore, the results of this model were also tested by implementing some *in vitro* experiments to prove that the real plasmodium is also capable to form proximity graphs such as letters of the alphabet. More specifically, various cases were tested where the plasmodium tried to recreate letters of the alphabet and here we present, for sake of readability and space, the cases of 'A' and 'O'. Twenty oat flakes, the nodes of the graph, were used in the experiment to create the 'A' letter. Eight oat flakes were placed in each one of the side edges, three nodes for the horizontal one and one node is placed at the top vertex (Fig. 17(a)). A small portion of plasmodium (c. 60mg) is placed on the top of each oat flake. Each plasmodium starts the expansion of its mass in every direction searching for new sources of nutrition (Fig. 17(b)). The plasmodium spreads out pseudopodia detecting relative locations of the closest sources of nutrients. Using chemotaxis, those pseudopodia sense the chemo-attractants released by food sources. When another FS is reached the relevant part of the plasmodium shrinks to a protoplasmic strand. This strand, also called a tube, connects the initial and newly acquired nutrients. In our model, the time $t = 0$ starts when the pseudopodia are expanded and have reached the food sources. Each plasmodium cell is represented by one agent in the CA model. The cells interact

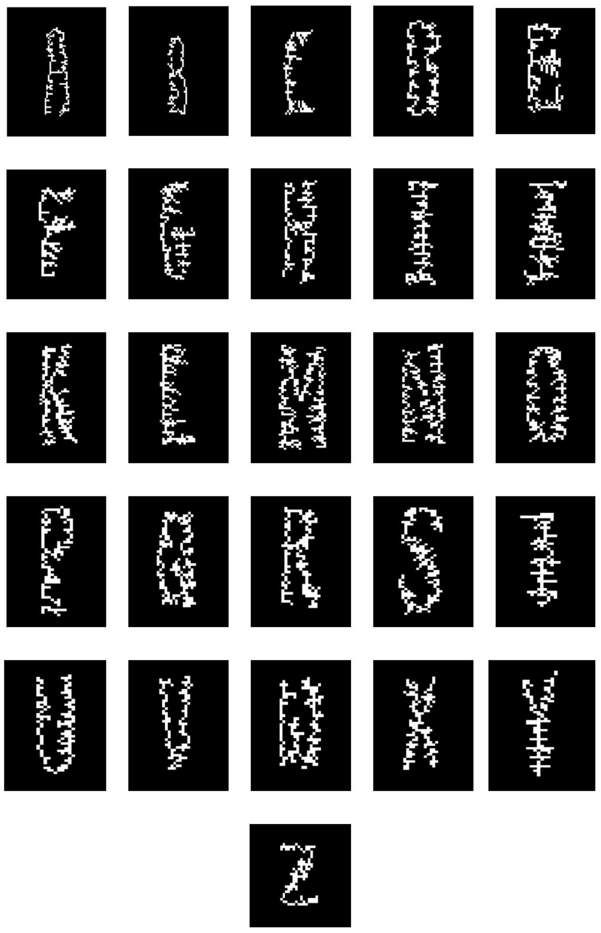

FIGURE 16. The letters from 'A' to 'Z' produced by the CA agent model.

with each other and they start creating a main protoplasmic tube between sources of nutrients. The graph that is created from this protoplasmic tube is the result of the experiment that is compared with the CA model's outcome. In Fig. 17(c) the plasmodium starts to create the desirable tube between sources of food forming the letter 'A'. In Fig. 17(d) the plasmodium has created strong tube connections between the food sites that form the edges of letter 'A' and tries to reach other food spots in Petri dish so to expand its mass in every direction.

The same experimental process was repeated to form the letter 'O' (Fig. 18. Sixteen oat flakes were placed in the petri dish in a circle with plasmodium on top of them. The experiment follows the same stages of evolution as the previous one. Finally, the plasmodium creates strong connection via tubes between food sources and the letter 'O' is formed. Both experiments similar results and there is a noise in the petri dish from other smaller tubes or centers of mass. However, these results are very close to the model's output which, as mentioned before, also does not produce clear paths between nodes to create the alphabet in every case as explained before. As a conclusion, it can be deduced that the experiments produce very close to the CA model results and can be used as a proof of concept.

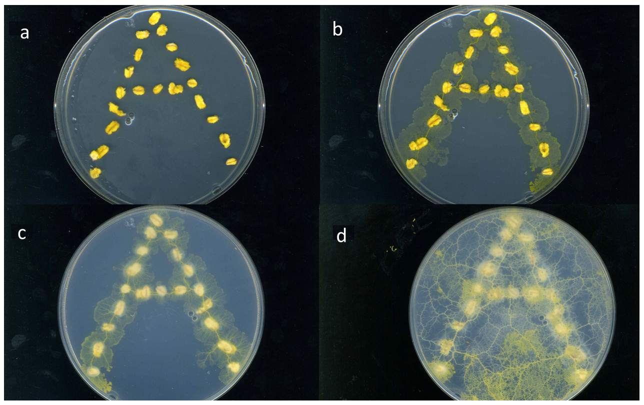

FIGURE 17. Approximation of letter 'A' by *Physarum polycephalum*. (a) The experiments start when the FSs with plasmodium are placed inside the petri dish. (b) Pseudopodia are spread using chemotaxis to find new sources of nutrients. (c) The mass of connected plasmodia is reshaped and shrank and it starts creating a main protoplasmic tube. (d) The letter 'A' is formed by the stronger tubes. After creating strong connections between FSs, Physarum expands almost in every site of petri dish to find new FSs.

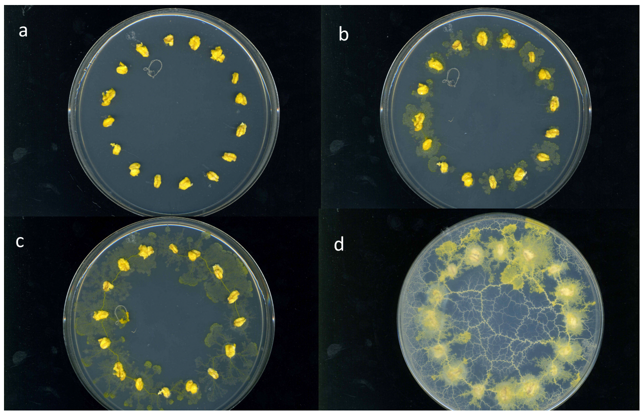

FIGURE 18. Approximation of letter 'O' by Physarum Polycephalum. (a) The FSs with plasmodium on top of them are placed inside the petri dish in a circle. (b) Pseudopodia search for new FSs. (c) When the pseudopodia reach the new FSs by following the chemo-attractants, they start forming a main protoplasmic strand. (d) The letter 'O' is formed. After creating strong connections between FSs, Physarum expands again in every site of petri dish to find new FSs.

VIII. CONCLUSIONS

The organism *Physarum Polycephalum* has recently attracted a wide range of scientists and engineers, because of its ability to find solution of spatially distributed computational problems. In the vegetative phase of its life, the plasmodium, as is referred, the slime mould shows non-trivial patterns of behavior when tries to create adaptive designs. This is achieved, because the plasmodium is capable to sensing the local concentrations of nutritional substances. Their presence seems to change the structure of some external places in the membrane and this causes changes to the internal substances of the organism.

There is no central control or any specified tissue in the organism. This feature is very convenient for the CA, a powerful parallel modeling tool. From the CA definition we know that the local rules can be considered as an alternative form of the microscopic reality, which supports the whole macroscopic behavior. There is no need to handle the macroscopic behavior of the system, since the local rules ensure that this will be produced by the interaction in the microscopic world without any central control. Furthermore, the simplicity of the organism's structure promotes the use of more simple models and CA rules.

We designed and proposed a novel model based on CA ABMs which has the ability to reproduce the biological experiment and create solid tubes between FSs. This model exploits the inherent parallelism of the CA and CA agents. We defined the local rules of the CA and the initial conditions which in this model are very important. A small change to the initial conditions can lead to successful or less successful experiments. This is enhanced if we consider that the model is based on stochastic rules and initial conditions. So, in order to test the efficacy of our model we tried to achieve reproducibility in the well known maze solving problem and travelling salesman problem. Furthermore, we tested the model's ability to form the shapes of the English alphabet's letters and combined its output with real organism experiments. In order to exploit the model's parallelism we implemented the tests in GPU and FPGA environment and repeated the tests multiple times. The results showed that the model achieved to mimic the experiments *in vitro* and accelerate them. As a result, a virtual lab describing the behavior of *Physarum Polycephalum* was created speeding up significantly the biological paradigm and its expected functionality when applied to various problems.

REFERENCES

[1] T. Nakagaki, H. Yamada, and Á. Tóth, "Intelligence: Maze-solving by an amoeboid organism," *Nature*, vol. 407, no. 6803, p. 470, 2000.

[2] A. Adamatzky, "Slime mold solves maze in one pass, assisted by gradient of chemo-attractants," *IEEE Trans. Nanobiosci.*, vol. 11, no. 2, pp. 131–134, Jun. 2012.

[3] J. Jones, R. Mayne, and A. Adamatzky, "Representation of shape mediated by environmental stimuli in physarum polycephalum and a multi-agent model," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 32, no. 2, pp. 166–184, 2017.

[4] N. Dourvas, M.-A. Tsompanas, G. C. Sirakoulis, and P. Tsalides, "Hardware acceleration of cellular automata Physarum polycephalum model," *Parallel Process. Lett.*, vol. 25, no. 1, 2015, Art. no. 1540006.

[5] J. Jones and A. Adamatzky, "Material approximation of data smoothing and spline curves inspired by slime mould," *Bioinspiration Biomimetics*, vol. 9, no. 3, 2014, Art. no. 036016.

[6] M.-A. I. Tsompanas, G. C. Sirakoulis, and A. I. Adamatzky, "Physarum in silicon: The Greek motorways study," *Natural Comput.*, vol. 15, no. 2, pp. 279–295, 2016.

[7] A. Adamatzky and J. Jones, "Road planning with slime mould: If Physarum built motorways it would route M6/M74 through newcastle," *Int. J. Bifurcation Chaos*, vol. 20, no. 10, pp. 3065–3084, 2010.

[8] V. Evangelidis, M.-A. Tsompanas, G. C. Sirakoulis, and A. Adamatzky, "Slime mould imitates development of Roman roads in the Balkans," *J. Archaeological Sci., Rep.*, vol. 2, pp. 264–281, Jun. 2015.

[9] B. Taylor, A. Adamatzky, J. Greenman, and I. Ieropoulos, "Physarum polycephalum: Towards a biological controller," *Biosystems*, vol. 127, pp. 42–46, Jan. 2015.

[10] V. S. Kalogeiton, D. P. Papadopoulos, and G. C. Sirakoulis, "Hey physarum! Can you perform SLAM?" *Int. J. Unconventional Comput.*, vol. 10, no. 4, pp. 271–293, 2014.

[11] J. G. Whiting, B. P. de L. Costello, and A. Adamatzky, "Transfer function of protoplasmic tubes of Physarum polycephalum," *Biosystems*, vol. 128, pp. 48–51, Feb. 2015.

[12] A. Adamatzky, "Slime mould electronic oscillators," *Microelectron. Eng.*, vol. 124, pp. 58–65, Jul. 2014.

[13] R. Mayne, M.-A. Tsompanas, G. C. Sirakoulis, and A. Adamatzky, "Towards a slime mould-FPGA interface," *Biomed. Eng. Lett.*, vol. 5, no. 1, pp. 51–57, 2015.

[14] A. Adamatzky, *Advances in Physarum Machines: Sensing and Computing With Slime Mould*, 1st ed. New York, NY, USA: Springer, 2016.

[15] J. Lee, C. Oettmeier, and H.-G. Döbereiner, "A novel growth mode of Physarum polycephalum during starvation," *J. Phys. D, Appl. Phys.*, vol. 51, no. 24, 2018, Art. no. 244002.

[16] G. Bretti and R. Natalini, "Numerical approximation of nonhomogeneous boundary conditions on networks for a hyperbolic system of chemotaxis modeling the Physarum dynamics," *J. Comput. Methods Sci. Eng.*, vol. 18, no. 1, pp. 85–115, 2018.

[17] K. Alim, N. Andrew, A. Pringle, and M. P. Brenner, "Mechanism of signal propagation in Physarum polycephalum," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 20, pp. 5136–5141, 2017.

[18] V. Ntinas, I. Vourkas, G. C. Sirakoulis, A. Adamatzky, and A. Rubio, "Coupled physarum-inspired memristor oscillators for neuron-like operations," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[19] Y. Liu, C. Gao, and Z. Zhang, "Simulating transport networks with a Physarum foraging model," *IEEE Access*, v9ol. 7, pp. 23725–23739, 2019.

[20] Z. Sun, Z. Chen, H. Hu, and J. Zheng, "Ship interaction in narrow water channels: A two-lane cellular automata approach," *Phys. A, Stat. Mech. Appl.*, vol. 431, pp. 46–51, Aug. 2015.

[21] Q. Chen and Y. Wang, "Cellular automata (CA) simulation of the interaction of vehicle flows and pedestrian crossings on urban low-grade uncontrolled roads," *Phys. A, Stat. Mech. Appl.*, vol. 432, pp. 43–57, Aug. 2015.

[22] M. Zamith, R. C. P. Leal-Toledo, E. Clua, E. M. Toledo, and G. V. P. de Magalhães, "A new stochastic cellular automata model for traffic flow simulation with drivers' behavior prediction," *J. Comput. Sci.*, vol. 9, pp. 51–56, Jul. 2015.

[23] T. Ghisu, B. Arca, G. Pellizzaro, and P. Duce, "An optimal cellular automata algorithm for simulating wildfire spread," *Environ. Model. Softw.*, vol. 71, pp. 1–14, Sep. 2015.

[24] D. I. Iudin, Y. D. Sergeyev, and M. Hayakawa, "Infinity computations in cellular automaton forest-fire model," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 20, no. 3, pp. 861–870, 2015.

[25] D. Scalise and R. Schulman, "Emulating cellular automata in chemical reaction–diffusion networks," *Natural Comput.*, vol. 15, no. 2, pp. 197–214, 2016.

[26] N. I. Dourvas, G. C. Sirakoulis, and A. Adamatzky, "Cellular automaton Belousov–Zhabotinsky model for binary full adder," *Int. J. Bifurcation and Chaos*, vol. 27, no. 6, 2017, Art. no. 1750089.

[27] N. I. Dourvas and G. C. Sirakoulis, "A inhibitor sensitive, collision based switching like transistor element using periodic traveling waves and cellular automata," *Int. J. Unconventional Comput.*, vol. 13, nos. 4–5, pp. 377–397, 2018.

[28] Y. Chen, N. Chen, Y. Wang, Z. Wang, and G. Feng, "Modeling pedestrian behaviors under attracting incidents using cellular automata," *Phys. A, Stat. Mech. Appl.*, vol. 432, pp. 287–300, Aug. 2015.

[29] R. da Silva, A. Hentz, and A. Alves, "Stochastic model of self-driven two-species objects inspired by particular aspects of a pedestrian dynamics," *Phys. A, Stat. Mech. Appl.*, vol. 437, pp. 139–148, Nov. 2015.

[30] E. Boukas, I. Kostavelis, A. Gasteratos, and G. C. Sirakoulis, "Robot guided crowd evacuation," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 2, pp. 739–751, Apr. 2015.

[31] A. Tsiftsis, I. G. Georgoudas, and G. C. Sirakoulis, "Real data evaluation of a crowd supervising system for stadium evacuation and its hardware implementation," *IEEE Syst. J.*, vol. 10, no. 2, pp. 649–660, Jun. 2016.

[32] V. S. Kalogeiton, D. P. Papadopoulos, I. P. Georgilas, G. C. Sirakoulis, and A. I. Adamatzky, "Cellular automaton model of crowd evacuation inspired by slime mould," *Int. J. Gen. Syst.*, vol. 44, no. 3, pp. 354–391, 2015.

[33] T. Giitsidis, N. I. Dourvas, and G. C. Sirakoulis, "Parallel implementation of aircraft disembarking and emergency evacuation based on cellular automata," *Int. J. High Perform. Comput. Appl.*, vol. 31, no. 2, pp. 134–151, 2017.

[34] P. Schmiedgen and S. Wiesenhütter, and J. R. Noennig, "Transferring functions of biological immune systems to communication processes in disasters using cellular automata," *Procedia Comput. Sci.*, vol. 35, pp. 1333–1341, Jan. 2014.

[35] J.-X. Liu, X.-F. Li, G. Han, N. Sun, and K. Du, "Cellular automata model for bacterial information sharing mechanism," in *Proc. 10th Int. Conf. Natural Comput. (ICNC)*, Aug. 2014, pp. 354–359.

[36] T. Shirakawa, H. Sato, and S. Ishiguro, "Construction of living cellular automata using the Physarum plasmodium," *Int. J. Gen. Syst.*, vol. 44, no. 3, pp. 292–304, 2015.
[37] M.-A. I. Tsompanas and G. C. Sirakoulis, "Modeling and hardware implementation of an amoeba-like cellular automaton," *Bioinspiration Biomimetics*, vol. 7, no. 3, 2012, Art. no. 036013.
[38] G. Fullstone, J. Wood, M. Holcombe, and G. Battaglia, "Modelling the transport of nanoparticles under blood flow using an agent-based approach," *Sci. Rep.*, vol. 5, p. 10649, Jun. 2015.
[39] J. M. G. de Durana, O. Barambones, E. Kremers, and L. Varga, "Agent-based modeling of the energy network for hybrid cars," *Energy Convers. Manage.*, vol. 98, pp. 376–386, Jul. 2015.
[40] I. Kucukkoc and D. Z. Zhang, "Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines," *Int. J. Adv. Manuf. Technol.*, vol. 82, nos. 1–4, pp. 265–285, 2016.
[41] O. A. Filatova and P. J. Miller, "An agent-based model of dialect evolution in killer whales," *J. Theor. Biol.*, vol. 373, pp. 82–91, May 2015.
[42] C. Tischer, J.-P. Zock, M. Valkonen, G. Doekes, S. Guerra, D. Heederik, D. Jarvis, D. Norbäck, M. Olivieri, J. Sunyer, C. Svanes, M. Täubel, E. Thiering, G. Verlato, A. Hyvärinen, and J. Heinrich, "Predictors of microbial agents in dust and respiratory health in the Ecrhs," *BMC pulmonary Med.*, vol. 15, no. 1, p. 48, 2015.
[43] J. Jones, "Characteristics of pattern formation and evolution in approximations of Physarum transport networks," *Artif. Life*, vol. 16, no. 2, pp. 127–153, Mar. 2010.
[44] J. Jones and A. Adamatzky, "Emergence of self-organized amoeboid movement in a multi-agent approximation of Physarum polycephalum," *Bioinspiration Biomimetics*, vol. 7, no. 1, 2012, Art. no. 016009.
[45] J. Jones and A. Adamatzky, "Computation of the travelling salesman problem by a shrinking blob," *Natural Comput.*, vol. 13, no. 1, pp. 1–16, Mar. 2014.
[46] J. Jones, S. Tsuda, and A. Adamatzky, "Towards Physarum robots," in *Bio-Inspired Self-Organizing Robotic Systems*. Berlin, Germany: Springer, 2011, pp. 215–251.
[47] J. Sweet, D. H. Richter, and D. Thain, "GPU acceleration of Eulerian–Lagrangian particle-laden turbulent flow simulations," *Int. J. Multiphase Flow*, vol. 99, pp. 437–445, Feb. 2018.
[48] A. G. Lewis and H. P. Pfeiffer, "GPU-accelerated simulations of isolated black holes," *Classical Quantum Gravity*, vol. 35, no. 9, 2018, Art. no. 095017.
[49] Y. Song, S. Yang, and J. Lei, "ParaCells: A GPU architecture for cell-centered models in computational biology," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 16, no. 3, pp. 994–1006, May/Jun. 2018.
[50] F. E. H. Pérez, N. Mukhadiyev, X. Xu, A. Sow, B. J. Lee, R. Sankaran, and H. G. Im, "Direct numerical simulations of reacting flows with detailed chemistry using many-core/GPU acceleration," *Comput. Fluids*, vol. 175, pp. 73–79, Sep. 2018.
[51] N. I. Dourvas, G. C. Sirakoulis, and P. Tsalides, "GPU implementation of physarum cellular automata model," in *Proc. AIP Conf.*, 2015, vol. 1648, no. 1, Art. no. 580019.
[52] G. T. Toussaint, "The relative neighbourhood graph of a finite planar set," *Pattern Recognit.*, vol. 12, no. 4, pp. 261–268, 1980.
[53] J. Nešetřil, E. Milková, and H. Nešetřilová, "Otakar Borůvka on minimum spanning tree problem translation of both the 1926 papers, comments, history," *Discrete Math.*, vol. 233, nos. 1–3, pp. 3–36, 2001.
[54] B. Delaunay, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, nos. 793–800, pp. 1–2, 1934.
[55] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," *Syst. Biol.*, vol. 18, no. 3, pp. 259–278, 1969.
[56] A. Adamatzky, "Developing proximity graphs by physarum polycephalum: Does the plasmodium follow the toussaint hierarchy?" *Parallel Process. Lett.*, vol. 19, no. 1, pp. 105–127, 2009.
[57] D. Applegate, R. Bixby, V. Chvátal, and W. J. Cook. (2006). *Concorde TSP Solver*. [Online]. Available: http://www. tsp. gatech. edu/concorde

NIKOLAOS I. DOURVAS received the Diploma degree in electrical and computer engineering and the M.Sc. degree from the Democritus University of Thrace (DUTh), Greece, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree. His current interests include cellular automata, modeling large scale systems, parallel programming, embedded systems, and bio-inspired algorithms.

GEORGIOS CH. SIRAKOULIS received the Diploma degree in electrical and computer engineering from the Democritus University of Thrace (DUTh), Greece, in 1996, and the Ph.D. degree in electrical and computer engineering from the Democritus University of Thrace, Greece, in 2001. In Diploma Thesis, he received a prize of distinction from the Technical Chamber of Greece (TEE). He has been a tenure Full Professor with the Department of Electrical and Computer Engineering, Democritus University of Thrace, since 2008. His courses lab activities are sponsored by ARM, Freescale, Xilinx, and Altera. He was also a Founding Member and the Vice President of the IEEE Student Branch of Thrace, from 2000 to 2001. He has served as a member for the EU IDEAS programme. He is a member of the IEEE Computer Society, of the Institute of Electrical Engineering (IEE), of the Association of Computing Machinery (ACM), of the International Society for Computational Biology (ISCB), and of the TEE. He is an Associate Editor of *Microelectronics Journal*, *Integration*, the *VLSI Journal*, the *Journal of Cellular Automata*, the *International Journal of Unconventional Computing*, the *International Journal of Parallel, Emergent and Distributed Systems*, *Parellel Processing Letters*, the IEEE Transactions on Computers, the IEEE Transactions on Nanotechnology. He is EUROPRACTICE representative for DUTh.

ANDREW I. ADAMATZKY is currently a Professor of unconventional computing and the Director of the Unconventional Computing Laboratory, Department of Computer Science, University of the West of England, Bristol, U.K. He authored seven books, mostly notable are *Reaction-Diffusion Computing*, *Dynamics of Crow Minds*, *Physarum Machines*, and edited 22 books in computing, most notable are *Collision Based Computing*, *Game of Life Cellular Automata*, *Memristor Networks*. He also produced a series of influential artworks published in the atlas *Silence of Slime Mould*. His research interests include molecular computing, reaction-diffusion computing, collision-based computing, cellular automata, slime mould computing, massive parallel computation, applied mathematics, complexity, nature-inspired optimisation, collective intelligence and robotics, bionics, computational psychology, non-linear science, novel hardware, and future and emergent computation. He is Founding Editor-in-Chief of the *Journal of Cellular Automata* and the *Journal of Unconventional Computing* and an Editor-in-Chief of the *Journal of Parallel, Emergent, Distributed Systems* and *Parallel Processing Letters*.