Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100607

A GPGPU Physarum Cellular Automaton Model

Nikolaos I. Dourvas, Georgios Ch. Sirakoulis and Philippos Tsalides*

Laboratory of Electronics, Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, 67100, Greece

Received: 30 Jan. 2015, Revised: 9 Dec. 2015, Accepted: 13 Feb. 2016
Published online: 1 Nov. 2016

Abstract: Scientists have been gaining inspiration from several natural processes and systems to find fine solutions in many complex hard to solve engineering problems for many years now. Nevertheless, most of these natural systems suffer from great amount of time to perform; thus, scientists are seeking for computational tools and methods that could encapsulate in a conscious way nature's genius, dealing at the same moment with time complexity. In this conquest, Cellular Automata (CA) proposed long time ago by John von Neumann, can be considered as a promising candidate. CA have the ability to capture the essential features of systems in which global complicated behavior emerges from the collective effect of simple components, which interact locally. These characteristics are immanent in many natural systems; namely *Physarum polycephalum*,an amoeba, is such a system. This simple organism presents the intelligence of finding effective solutions to demanding engineering problems such as shortest path(s) problems, various graph problems, evaluation of transport networks or even robotic control. In this paper, we move forward by taking advantage of a Graphical Processing Unit (GPU) and the Compute Unified Device Architecture (CUDA) programming model, to make use of the CA inherit parallelism when biomimicking the behavior of *P. polycephalum* in maze, providing the ability to find the minimum path between two spots. In this way we are able to produce a virtual easy-to-access lab speeding up significantly the biological paradigm when modeled by CA implemented in General Purpose computing on Graphics Processing Units (GPGPU) environment.

Keywords: physarum polycephalum, cellular automata, GPU, CUDA

1 Introduction

Physarum polycephalum is a slime mold. The plasmodium of Physarum is a large amoeba-like cell consisting of a dendritic network of tube-like structures. It has been observed that it has the ability to change its shape as it crawls over a plain agar gel, and, moreover, if food is placed at two different points, it will put out pseudopodia that connect the two food sources (FSs) [1]. The last observation enables us to think that alike other living organisms giving inspiration to scientists for solving hard complex problems [2], *Physarum* or commonly known as true slime mould, could be considered as such [3]. 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 and demonstrated complicated and robust computing capacity when it confronted solving maze problems [1,6]. However, its abilities are not merely that limited. After that, there was a burst of research on this simple organism since a lot of researchers successfully utilized it as an unconventional computing material that exposed a great range of its computational abilities to spatial representations of various graph problems, combined optimization problems, construction of logic gates and logical machines [3,8,10,9,11,12]. Plasmodium, which is a vegetative phase of *Physarum*'s life cycle, due to its simplicity, extreme easiness to be cultivated and handled, and the exhibition of remarkably interesting foraging behavior, has been also successfully used in the field of robotic control [13], robotic amoebic movement [14] and for robotic Simultaneous Localization and Mapping (SLAM) [23]. As a disadvantage, the experiments on a living organism last a lot of hours or more specifically some days to finish [25]. So the necessity of modeling its behavior as precise and as fast as possible is the key for further exploitation of slime mould unconventional computing abilities. There is a variety of modeling approaches because there is no single model that can describe exactly the behavior of *Physarum*, considering only the plasmodium stage. Those models use also different modeling tools. The bibliography presents some purely spatial Cellular

* Corresponding author e-mail: gsirak@ee.duth.gr

Automaton (CA) models, [15,16,17], a mathematical representations of flux canalization [18], oscillatory behavior [19,20], a two-variable Oregonator model of Belousov-Zhabotinsky (BZ) medium [21] and a multi-agent model applied to a path planning problem [22].

In this paper, we make use of CA to biomimick slime mould foraging behaviour and thus, the computing abilities of the plasmodium of *P. polycephalum* to find shortest path in a maze. CA are a very elegant computing model that dates back to John Von Neumann [26] and Konrad Zuse [27]. CA are models of physical systems, where space and time are discrete and interactions are local. As such computational systems can be applied to many real problems in physics, chemistry, biology and also to computational or artificial problems [28,29,30, 31]. The last decades, a wide variety of CA applications have been proposed on several scientific fields, such as simulation of physical systems, biological modeling involving models for self-reproduction, biological structures, image processing, semiconductor fabrication processes, crowd evacuation, computer networks and quantum CAs [32,33,34,35,36,37,38,39,40]. These problems are described in terms of CA, spatially by an 1-d, 2-d or 3-d array of cells and a local rule, which is usually an arbitrary function that defines the new state(s) of its CA cell depending on the states of its CA neighbors. The CA cells can work in fully synchronous and parallel manner updating their own state. It is clear that the CA approach can be considered consistent with the modern notion of unified space time, where, in computer science, space corresponds to memory and time to processing unit. In analogy, in CA, memory (CA cell state) and processing unit (CA local rule) are inseparably related to a CA cell [41,42]. Therefore, the resulting CA model of slime mould presented in this paper is massively parallel and, consequently, can be considered in a straightforward manner an ideal candidate to be implemented in a Graphical Processing Unit (GPU). The idea is that even if modern computers offer sufficient processing power to handle most of the analysis that several complex phenomena require, in several cases it is of utter importance to increase the performance of modeling procedures to take the results faster. A method to speed-up the execution of an algorithm, which uses information data in parallel, is to use the potential of available and increasingly popular GPUs. Today's graphics cards in computing performance monifoldingly prevail over Central Processing Units (CPUs). Since 2006, from the appearance of the NVIDIA G80 type chip, they can be programmable to general computing tasks [50]. This was the first card, which worked on newly developed general processing units instead of the special vertex and pixel shader.These new processors can be interpreted as simply, scalar ALU-s which execute the same simple instructions parallel on each data of the input stream thus creating the output stream. This single instruction multiple thread (SIMT) architecture results in giant computing performance. GPUs have now affordable cost and can execute high performance scientific computing on personal computers, and the good price per performance ratio makes them ideal option. As a result, the application of GPUs encouraged software engineers to utilize the advantages of their CA models with enlarged functionality [43,44,45,46,47]. There have been also researches who aim to investigate how to use the graphics hardware for general computing ability in biological CA based models.[48]. Furthermore, among others, the Compute Unified Device Architecture (CUDA)- a program developing environment developed by NVIDIA-makes possible to program this device on high level programming languages such as C or C++ boosting the performance of corresponding software and enhancing the proposed usability [49].

In the following sections, we give a short description of *Physarum polycephalum* as an organism as well a small literature survey on the experiments which have been published so far. In Section 3, we analyze the mathematical model on which all the *Physarum* theory is based in order to solve the shortest path problem of the maze. After that, in Section 4 we make a short description of Cellular Automata and we give some indicative published paradigms to prove how important modeling tool is for the research of *Physarum polycephalum*. The CA model that is used in this paper and tries to describe effectively the behavior of the plasmodium in a maze, is analyzed in Section 5. Finally, the algorithm used in order to parallelize the proposed GPU model biomimicking slime mould's behaviour and speed-up the simulations using the GPU and the CUDA programming model is presented in Section 6. Conclusions and further future work are drawn in Section 7.

2 Physarum Basics

Physarum as every biological system adapts to its environment. Its aim is to balance the cost of producing an efficient network with the consequences of even limited failure in a competitive world. Many years of evolutionary selection have passed and these biological systems have survived, so they have reached a great balance between cost, efficiency and resilience. The observation of those systems' behavior led to useful approaches to many complicated problems solving such as graph problems, neural networks, genetic algorithms, swarm intelligence, etc. [52,51,2]. In the same manner using insights gained by the observation of laboratory experiments with the plasmodium of *P. polycephalum* has triggered the scientific community to further explore the abilities of *Physarum* as a novel unconventional computing substrate.

Physarum belongs to the species of order *Physarales*, subclass *Myxogastromycetidae*, class *Myxomycetes* and division *Myxostelida*. It is a large, acellular or multi-headed slime mould. *P.polycephalum* passes

through many phases in its complex life cycle. Plasmodium is a "vegetative" phase, a single cell with myriad of diploid nuclei. It is visible to the naked eye and looks like an amorphous yellowish mass with networks and protoplasmic tubes. The plasmodium behaves and moves as a giant amoeba. It feeds on bacteria, spores and other microbial creatures and micro-particles. When foraging for this food, the plasmodium propagates towards sources of food particles, surrounds them, secretes enzymes and digests the food [53]. More specific, it develops a tubular network linking the discovered food spots (FSs) through direct connections. In order to reduce the overall length of the connecting network it creates additional intermediate junctions (Steiner points). The characteristics of the substrate or the FS, above which the plasmodium forages, can play a key role on the growth of the amoeba [54]. There are also some constraints such as physical barriers or light regime which can limit the foraging behavior of plasmodium in specific places. It is very important to remember that the organism is not explicitly trying to solve computational problems. So the idea served by several researchers was how to take advantage of *Physarum*'s to survive in order to solve complex problems. In such a fashion, the constraints and tools mentioned before are used to control the plasmodium and fit well to the under study problem. Therefore, the investigation of rules that lead to network formation can be achieved by tuning with the above parameters. For example, *Physarum* can find the shortest path through a maze, or connect different arrays of FSs in an efficient manner with low total length, short average minimum distance between pairs of FSs and with high degree of fault tolerance to instant or accidental disconnections.

Some of the most relevant works to the maze solving and network formation problem are presented below, in order to point out the research interest the *Physarum polycephalum* provides. First, Nakagaki *et al.* [1] were the first to observe that the plasmodium of the slime mould changes its shape as it crawls over a plain agar gel. If food is placed in two certain spots, it puts out pseudopodia that connect those food spots. The most interesting part is that the plasmodium had the ability to find the minimum-length solution between two points in a labyrinth. This happens because *Physarum* reduces its mass, from the paths of the maze that is far from the minimum distance, and strengthens its tubes that belong to the minimum distance. Its goal is to survive without having to connect FSs with large paths. The result of the *Physarum*'s solution to the maze problem with 2 FSs is shown in Fig. 1. It is very important to clarify that in this study the Nakagaki's maze topology was used but the approach of the plasmodium's movement is different.[6] That exact behavior gave inspiration to the researchers to discover more properties of this organism synonymous to unconventional computing principles and take advantage of them in many more applications. So, after their first foundation, Nakagaki *et al.* [55] moved forward and found that the geometry of the network, created by the plasmodium depended on the positions of the FSs. The statistical analysis showed that the network geometry met the multiple requirements of a smart network. There requirements are short total length of tubes, close connections among all the branches (a small number of transit food sites between any two food sites) and tolerance of accidental disconnection of the tubes. That led to the assumption that *Physarum* can provide better solution to the problem of network configuration than the Steiner's minimun tree solution. The organism derives the maximum of nutrient in the minimum of time.So all these conclude to the fact that P.polycephalum can be used to solve complex problems.

Adamatzky in [6] proposed another approach of the same problem. The main difference can be found in the initial conditions. Adamatzky placed the plasmodium in one place of the maze and, simultaneously, placed one FS in another place of the maze, before the plasmodium covers all the maze. The biological experiments show that the plasmodium spreads its pseudopodia trying to reach the food. Simultaneously, the food, releases the chemo-attractants to any direction in the maze. When the plasmodium finds those chemo-attractants, it follows them to the source food forming the minimum distance path between its initial site and the food site. So the plasmodium solves the maze in one pass because it is assisted by a gradient of chemo-attractants propagating from the target food. This approach, is modeled in this paper.

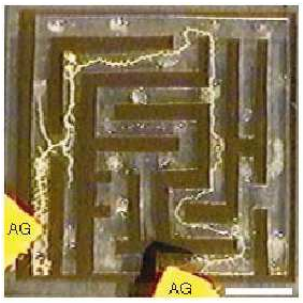

Fig. 1: Minimum distance between two FSs in a maze by *Physarum polycephalum* [1].

In the view of the foregoing, the algorithms inspired by plasmodium, have some features of interest embodied in unconventional computing and different from those observed in the classical computational logic algorithms. These *Physarum*-inspired algorithms are bottom-up, decentralized approaches that make use of simple set of conditions and rules, while they attempt to solve a complex problem by iteratively applying these rules. The most important feature, which is commonly used, is that plasmodium's algorithms are mainly based in local

interactions which can lead to very complicated behaviors.

In such a sense, *Physarum* has been also used on representations of various graph problems. Adamatzky in [7] managed to address the novel issues of executing graph optimization tasks on distributed simple growing biological systems. More specifically, the plasmodium is used as experimental computing substrate to approximate spanning trees. Points of given data sets are represented by positions of nutrient sources, while a plasmodium is placed on one of the data points. The results showed that plasmodium developed and spanned all sources of nutrients, connecting them by protoplasmic strands. The protoplasmic strands represent edges of the computed spanning tree. The practicality is that those techniques can be used in design and development of soft bodied robotic devices, including gel-based robots, reconfigurable massively robots and hybrid wet-hardware robots [3]. Furthermore, In [63] Shirakawa et al. experimentally demonstrated that both Voronoi diagram and its dual graph Delaunay triangulation are simultaneously constructed, for specific conditions, in cultures of plasmodium. Every point of a given planar data set was represented by a tiny mass of plasmodium. The plasmodium spread from the initial locations but, it stopped spreading when they encountered plasmodia originated from different locations. Space loci not occupied by the plasmodia represent edges of Voronoi diagram of the given planar set. At the same time, the plasmodia originating at neighboring locations formed merging protoplasmic tubes, where the strongest tubes approximate Delaunay triangulation of the given planar set. The problems were solved by plasmodium only for limited data sets, however the results presented lay a sound ground for further investigations.

In 2010, Tero *et al.* [68] compared the actual rail network in Japan with a *Physarum* network consisted by 36 FSs that represented the geographical locations of cities in Tokyo area. The *Physarum* was planted on Tokyo and from there started its foraging behavior and exploration for FSs until it filled much of the available land space. Then the organism started to concentrate on the FSs by thinning out the network to leave a subset of larger interconnecting tubes as shown in Fig. 2. The topology of many *Physarum* networks appeared similar to the rail network. The conclusion was that *Physarum* networks showed characteristics similar to those of the rail network in terms of cost, transport efficiency and fault tolerance.

Furthermore, Adamatzky in his book [8] describes the ability of *Physarum* to mimic and evaluate real transport networks. In more details, Adamatzky and Jones were the first who proposed the evaluation of the ability of *Physarum* to approximate a road network and the actual man-made road networks [56]. First they applied the solution on UK road networks and afterwards Adamatzky *et al.* [57] applied it to Mexico networks. The results showed that the network of protoplasmic tubes, developed

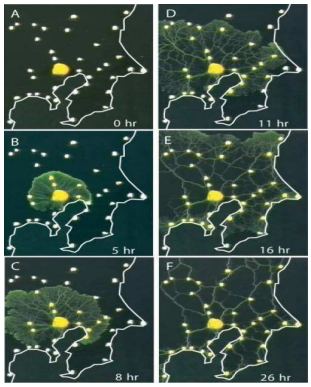

Fig. 2: Simulation of the Tokyo railway by *Physarum polycephalum* [68].

by plasmodium, matches at least partly the network of man-made transport arteries. Some parameters such as the shape of a country and exact spatial distribution of urban areas can play key role in determining exact structure of plasmodium network. Then, Adamatzky and Alonso-Sanz [58] attempted to find out how close plasmodium of *P. polycephalum* approximates man-made motorway networks in Spain and Portugal and what are the differences between existing motorway structure and plasmodium network of protoplasmic tubes. They cut agar plates in a shape of Iberia peninsula, placed oar flakes at the sites of major urban areas and analyzed the foraging network developed. In the same way, many other countries' transport networks like Germany, USA, Canada, etc. have been evaluated with the help of *Physarum* [59,60,61,62].

In [4], Jones used a particle model of slime mould and demonstrated experiments which indicated that path planning may be performed by morphological adaptation. More specific, he demonstrated simple path planning by a shrinking blob of virtual plasmodium between two attractant sources within a polygonal arena. He presents the subsequent selection of a single path from multiple options. To create this path, he used nutrients to attract the shrinking blob or hazardous stimuli (light irradiation, repellents, or warm regions) to create obstacle avoidance or collision-free paths. Moreover, Jones and Adamatzky used the same particle model to demonstrate a simple unconventional computation method to approximate the Euclidean TSP [5]. The shrinking blob was placed over a set of data points projected into the lattice (TSP city locations), and the blob was reduced in size over time. As the blob shrinked, it morphologically adapted to the configuration of the cities. The shrinkage process automatically stopped when the blob no longer completely covered all cities.

Some more interesting aspects of *Physarum* can be found in [64], where Shirakawa and Gunji tested for the presence of emergent properties in a biological system

using the simplest biological entity of the plasmodium. They let two plasmodium networks within a single cell interact with each other, and observed how the intracellular interaction affected the morphologenesis of the plasmodium networks. They found that the two networks developed homologous morphology. Moreover, Shirakawa et al. tested the presence of memory and learning ability in the plasmodium [65]. They performed an associative learning experiment using the unicellular organism. The plasmodium in this experiment seemed to acquire a reversed thermotactic property, a new preference for the lower temperature. The result implied a possibility of unicellular learning, though in a preliminary way. Finally, Shirakawa and Sato, based on results from previous associative learning experiments using the *Physarum* plasmodium, constructed a gene regulatory network model of unicellular learning [66]. The model demonstrated that, in principle, unicellular learning can be achieved through the cooperation of several biomolecules.

The slime mould, as a living substrate, does not halt its behavior when a task is solved but often continues foraging the space thus masking the solution found. The calculation of the termination time of an experiment modeled by virtual slime mould is a very complicated problem. At the beginning of computation the slime mould explores the space in order to detect the gradients of repellents and attractants. In this phase it generates less compressible patterns. After those gradients are detected the slime spans data sites with its protoplasmic network and retracts scouting branches. In this phase it generates more compressible patterns. Therefore, Adamatzky and Jones proposed the use of temporal changes in compressibility of the slime mould patterns as indicators of the halting of the computation [67].

3 Mathematical model of *Physarum polycephalum*

In this section a brief introduction in the basic principles of mathematical modeling of *Physarum* is provided. From a historical point of view, Tero and Nakagaki [69] proposed a mathematical model for the adaptive dynamics of the transport network in the true slime mold *P. polycephalum*. Their goal was to extract a mathematical algorithm to depicture this natural computation. Two empirical rules can describe the changes in the tubular structure of the plasmodium. The first one is that open-ended tubes are likely to disappear, and the second, when two or more tubes connect to the same two food spots the one which follows the greater distance tends to disappear. So the authors attempted to reproduce these rules in their mathematical model. After that, they applied it to the navigation problems posed by a complicated road map and a large labyrinth. Then, in 2008, Tero *et al.* [70] used the aforementioned model to develop a wide variety of network shapes ordinary observed in real experiments with three, four, six and more FSs, respectively. The output was that the model also reproduced the actual situation and with specific tuning of its parameters can lead to an algorithm for a Steiner solver problem.

But, before the mathematical model proposed by Tero [69] is revisited, let's examine the way that the organism solves a simple labyrinth. First, the plasmodium inundates the whole maze as it is shown in Fig. 3*a*. Then, in the presence of two FSs in two sites of the maze, the tubular network becomes more visible and the unnecessary tubes start to disappear (Fig. 3*b*). Finally, the tube with the minimum distance remains as shown in Fig. 3*c*. This labyrinth is the basic example, which is commonly used by many *Physarum* models to prove their validity and efficacy. The same labyrinth will also be used in this research.

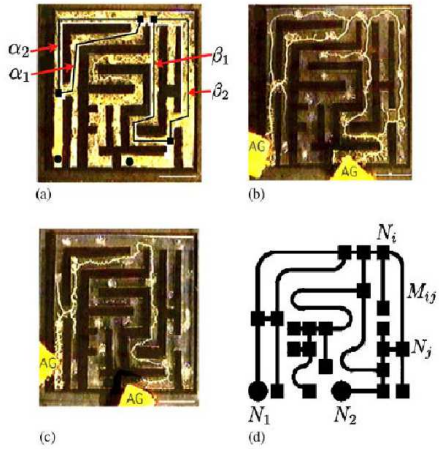

Fig. 3: Different phases in the procedure to the solution of the maze by *Physarum polycephalum*. (a) Initial state, (b) Intermediate state, (c) Final state, (d) Graphical representation of the maze [1].

The initial position of the tubular network is shown in the graph of Fig. 3*d*, where every edge represents a specific place of the network. The two special nodes representing the FSs are N_1 and N_2. The other nodes of the maze are N_3, N_4, etc. One of the two special nodes is assumed to be the source and the other the end. An edge between nodes N_i and N_j is called $M_{i,j}$.

The variable $Q_{i,j}$ is used to define the flow through the edge $M_{i,j}$, from the node N_i to the node N_j. So, the flow is defined by the equation 1.

$$Q_{i,j} = \frac{\pi a_{i,j}{}^4}{8\kappa} \frac{p_i - p_j}{L_{i,j}}, \tag{1}$$

where $L_{i,j}$ and $a_{i,j}$ is the length and the radius, respectively, of the tube which corresponds to the edge

$M_{i,j}$, κ is a constant of the mixture and p_i is the pressure on the node N_i. By setting $D_{i,j} = \pi {a_{i,j}}^4/8\kappa$, as the conductance of the edge, the equation 1 results to equation 2 as follows:

$$Q_{i,j} = \frac{D_{i,j}}{L_{i,j}}(p_i - p_j). \quad (2)$$

We assume as zero the capacity of every node and because of the principle of mass conservation we take the following equation 3:

$$\sum Q_{i,j} = 0, (j \neq 1,2). \quad (3)$$

But for the special nodes, we have the equation 4 as follows:

$$\sum Q_{i,1} + I_0 = 0, \sum Q_{i,2} - I_0 = 0, \quad (4)$$

where I_0 is the flow, which is going into the source node. This value is assumed as constant in the described model meaning that the total flow has a constant value during the whole procedure.

The experiments show that the tubes with great amount of flow are strengthened, while those with less flow are eventually weakened. To describe this adaptation of the radius of every tube, it is assumed that the conductance $D_{i,j}$ changes with time according to the flow $Q_{i,j}$. The equation 5 mentioned below has been proposed for its calculation.

$$\frac{d}{dt}D_{i,j} = f(|Q_{i,j}|) - rD_{i,j}, \quad (5)$$

where r is the attenuation rate of the tube. The equation 5 implies that the conductance tends to disappear if there is no flow across the length of the edge, whereas it is strengthened in the opposite situation. As it is natural, f is a monotonic increasing continuous function which satisfies the condition $f(0) = 0$. Note that the length of the edges, $L_{i,j}$, stays constant during the procedure of adaptation unlike the $D_{i,j}$.

In the simulations, the graph of fig. 3*d* is used, as mentioned above. The exponential equation $f(Q) = Q^\mu$ is used, where the variable μ takes positive values and two examples are given. The first one shows the results for values of μ greater than one and the second for values of μ less than one. The initial values of $Q_{i,j}$ are set randomly and take values between [0.5, 1.0]. In fig. 4 the results for $\mu > 1$ are also shown. More specifically,the evolution of the system, through the graphical representation of the conductance versus time for all the edges is given. As is is clear there is fast disappearance of the tubes that are not necessary for the calculation of the minimum distance. After some time, the paths α_1 and β_1 dominate. This happens regardless the initial conditions. But, if the initial value of edge β_1 is lower than 0.5 (fig. 4*c*,*d*), then the edge β_2 is obvious that stands until the end.

In fig. 5 is obvious that for greater values of exponent μ, the choice between competitive edges is faster. But, it is not certain that the minimum distance is going to be chosen ($a_1 - b_1$). In fig. 6 the results are presented for values $0 < \mu < 1$. It seems that all the four edges survive at the end of the experiment, but those, which correspond to the minimum distance, have the greater conductance.

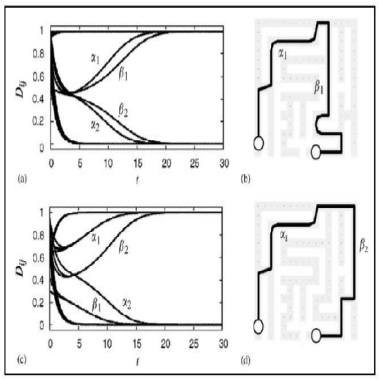

Fig. 4: Results of the mathematical model for $\mu = 1.2$. (a) Graphical representation of the conductance $D_{i,j}$ versus time. (b) Solution of the maze. (c) Graphical representation of the conductance $D_{i,j}$ versus time, (d) and solution of the maze with the difference that the conductance of the path β_1 is set lower than 0.5 [69]

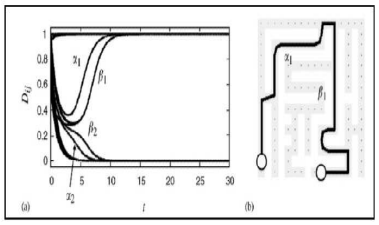

Fig. 5: Results of the mathematical model for $\mu = 2$. (a) Graphical representation of the conductance $D_{i,j}$ versus time. (b) Final solution for the maze [69].

It came apparent through the continuous experimentation of several researches with the specific organism that *Physarum* seems to have some kind of intelligence of emergent computation that can be formulated with the help of similar mathematical models. This ability is becoming apparent in nature in many different situations and, what is most important, successfully provides respective solutions to complex problems. Nevertheless, till now there is no single model that completely encapsulates *Physarum*'s behavior and in the vast majority of the published models only the plasmodium stage of its life is taken into account. Current attempts at modeling *Physarum*'s behavior try to simplify this huge task by compartmentalizing the different

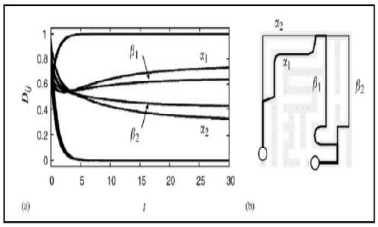

Fig. 6: Results of the mathematical model for $\mu = 0.9$. (a) Graphical representation of the conductance $D_{i,j}$ versus time. (b) Final solution for the maze [69].

behaviors of the organism under different situations, i.e. modeling the mechanisms of growth, the movement, the internal oscillations or the network adaptation.

4 Cellular Automata Basics

CA are models of physical systems, where space and time are discrete and interactions are local. They achieve that because they combine the use of memory (CA cell state) in order to save the information and the processing unit in order to process the information stored. They can capture the essential features of systems, where global behavior arises from the collective effect of simple components, which interact locally. In addition, they can handle complex boundary and initial conditions, inhomogeneities and anisotropies [30]. These CA characteristics are very convenient for describing the behavior and the dynamics of a biological organism such as *Physarum polycephalum*.

From a mathematical point of view, a CA consists of a regular uniform n-dimensional lattice (or array), usually of infinite extent. At each site of the lattice (cell), a physical quantity takes on values. This physical quantity is the global state of the CA, and the value of this quantity at each site is its local state. Each cell is restricted to local neighborhood interaction and, as a result, it is uncapable of immediate global communication [26]. The neighborhood of a cell is taken to be the cell itself and some (or all) of the immediately adjacent cells. The states at each cell are updated simultaneously at discrete time steps, based on the states in their neighborhood at the preceding time step. The algorithm used to compute the next cell state is referred to as the CA local rule. Usually, the same local rule applies to all cells of the CA in each time step simultaneously. This characteristic leads to synchronous dynamics and promotes parallel approaches in implementations. The rule is homogeneous which means that it does not depend explicitly on the cell position $\mathbf{r}$. But CA theory gives the option introducing spatial inhomogeneities by defining a specific value on some $C_j(\mathbf{r})$ in some given locations of the lattice. This is very helpful because a new rule can be applied in those marked cells. Usually, the memory of our cells hold their previous state and in the next step they hold the new value which derives from the rule. But in some occasions, it is important to have a longer memory and to introduce a dependence of the states at time $t-1, t-2, \ldots, t-k$. This situation is already included in the definition. The only thing that someone has to do is to copy the previous state in the current state.

In general a CA requires [71]:

1. a regular lattice of cells covering a portion of a d-dimensional space;
2. a set $C(\mathbf{r},t) = (C_1(\mathbf{r},t), C_2(\mathbf{r},t), ..., C_m(\mathbf{r},t))$ of variables attached to each site $\mathbf{r}$ of the lattice giving the local state of each cell at the time $t = 0, 1, 2, ...$;
3. a rule $R = (R_1, R_2, ..., R_m)$ which specifies the time evolution of the states $C(\mathbf{r},t)$ in the following way: $C_j(\mathbf{r}, t+1) = R_j(C(\mathbf{r},t), C(\mathbf{r}+\delta_1,t), C(\mathbf{r}+\delta_2,t), ..., C(\mathbf{r}+\delta_q,t))$, where $\mathbf{r}+\delta_k$ designate the cells belonging to a given neighborhood of cell $\mathbf{r}$.

The state of the a cell at time step (t+1) is computed according to R. R is a function of the state of this cell at time step (t) and the states of the cells in its neighborhood at time step (t). Regarding the two–dimensional CA ($n = 2$), there are two fundamental types of neighborhoods that are mainly considered: *von Neumann* neighborhood, which consists of a central cell and its four geographical neighbors north, west, south and east, resulting in a diamond shaped neighborhood and can be used to define a set of cells surrounding a given cell (x_0, y_0). Equation 6 defines the Von Neumann neighborhood of range r.

$$N^N_{(x_0,y_0)} = \{(x,y) : |x-x_0| + |y-y_0| \leq (r)\} \qquad (6)$$

.

For a given cell (x_0, y_0) and range r, *Moore* neighborhood, that consists of the same cells with the *von Neumann* neighborhood together with the four other adjacent cells of the central cell (the northwester, northeaster, south–east and south west cells), can be defined by the following formula:

$$N^M_{(x_0,y_0)} = \{(x,y) : |x-x_0| \leq (r), |y-y_0| \leq (r)\} \qquad (7)$$

.

In most practical applications, when simulating a CA rule, it is impossible to deal with an infinite lattice. The system must be finite and have boundaries. Clearly, a site belonging to the lattice boundary does not have the same neighborhood as other internal sites. In order to define the behavior of these sites, neighborhood is extending for the sites at the boundary, thus leading to various types of boundary conditions such as periodic (or cyclic), fixed, adiabatic or reflection.

In order to mimic the *Physarum*'s behavior some CA models are also presented. Initially, CA like models were presented by Gunji and his colleagues, who showed that

their cell model, namely CELL [16], which is moving like an amoeba, can form an adaptive network to solve a maze, the Steiner minimum tree problem and a spanning tree problem. In [17] Gunji *et al.* introduced the idea of decreases and increases in the number of cells for network formation. They revised their model based on the transportation of the "vacant-particle" by implementing a decrease of the number of cells at FSs, revealing the attraction of protoplasm toward food stimuli. In 2012, Tsompanas and Sirakoulis [15] proposed a CA model that is based on local interactions and as a global behavior attempts to simplify and reproduce the diffusion equation of the mass and the diffusion equation of the chemo-attractants of FSs. Then they used their model in order to find the minimum path between two FSs in a labyrinth. The results were very impressive, because the *Physarum* model achieved to reproduce the experimental results but with minimum algorithmic and computational complexity. In 2014, Tsompanas *et al.* [72] used their model in order to reproduce the main Greek motorways. *Physarum* model achieved to reproduce the actual motorways and it also foresaw the construction of another basic Greek motorway which is currently under construction. Moreover, Tsompanas *et al.* has verified through an evolutionary approach the proposed model was verified on our previously published results [8] on imitation of man-transport networks in several countries with living *P. polycephalum* [25]. In all the examined cases, the corresponding CA results were meticulously compared with proximity graphs and the graphs produced by the plasmodium, and they sufficiently reproduced the *P. polycephalum*'s recorded behavior. In 2014, Kalogeiton *et al.* [23] proposed an innovative approach to tackle with the Simultaneous Localization and Mapping (SLAM) task. In particular, a fully autonomous robot, equipped only with an omnidirectional camera explored and mapped successfully an indoor unknown terrain by adopting the behavior of *Physarum polycephalum*. The obtained results were compared to the corresponding ones produced by the random movement algorithm as well as by an exhaustive search algorithm. In all the examined cases, the results revealed the superiority of the proposed method. Finally, Shirakawa *et al.* [24] created an experiment in which a cellular automata-like system was constructed using the living cell. They analyzed the exploratory behavior of the plasmodium by duplicating the experimental results in the simulation models of cellular automata. As a result, it was revealed that the behavior of the plasmodium are not reproduced by only local state transition rules and the reproduction demands a kind of historical rule setting.

4.1 GPU and CA combination

In the last few years were some early attempts to take advantage of the benefits of CA in the physical simulations and the exploitation of their parallel nature by using the GPUs. Many researchers have combined CA with GPU in different models and simulations. For example, Mroz *et al.* tried to compare the possibilities of using the GPGPU in continuous and discrete crowd dynamics, in order to simulate outdoor or large area pedestrian movement, and to make conclusions on the applicability of GPUs in engines of professional crowd simulations [44]. The proposed discrete model was basically a CA based model. In [45], Quesada-Barriuso *et al.* presented that a watershed algorithm based on a CA is a good choice for the late GPU architectures, especially when the synchronization rules are relaxed. In particular, they proposed a block-asynchronous computation strategy that maps the CA on the thread blocks of the GPU, which leads to an efficient exploitation of the memory hierarchy of the GPU. The method was also tuned to be applied to 3D volumes. The high speedups indicated the potential of this kind of algorithm for new architectures based on hundreds of cores. Campos and his colleagues proposed an electro-mechanical simulator of the cardiac tissue in [46]. The main feature was the low computational cost for real-time simulations. They used CA and mass-spring systems to model the cardiac behavior and they parallelized the code to run in GPU with CUDA. The result was a faster simulator compared to the existing partial differential equations simulators. Three fire propagation CA models that were programmed in CUDA were presented in [47]. The results, which were compared against the serial ones on CPU, achieved a speed-up of over 200 times; thus the simulation results were faster that real-time capabilities and may be useful for fire fighting methodologies.

5 Proposed *Physarum* CA model

In this Section we present the proposed CA model. We consider the biological experiment where the plasmodium was starved and then introduced into a specific site of the maze. Simultaneously, a FS which produces chemo-attractants is placed in another site of the maze. These special sites in the maze are the same as in fig. 1. In order to model the aforementioned *Physarum* experimentation with CA in two-dimensions, the area is divided into a matrix of identical squares and each square of this surface is represented by a CA cell. The type of neighborhood that was used in this CA model was chosen to be the Moore neighborhood (equation 7), for computational reasons. The state of the i, j cell at time t, defined as $C^t_{i,j}$ is arrives from the following equation:

$$C^t_{i,j} = \{CellType_{i,j}, Chem^t_{i,j}, Dir^t_{i,j}, Plasm^t_{i,j}, MinTube^t_{i,j}\}, \quad (8)$$

where $CellType_{i,j}$ is a two-bit variable, which indicates the type of area of the corresponding i, j cell. There are four possible values of $CellType_{i,j}$:

1. $CellType_{i,j} = ''00'$ which means that this cell is considered as a free one, in which the plasmodium could move.
2. $CellType_{i,j} = ''01'$ which means that this cell represents the area of the initially placed FS
3. $CellType_{i,j} = ''10'$ which means that this cell represents the area of the initially placed plasmodium.
4. $CellType_{i,j} = ''11'$ which means that this cell is considered as a physical obstacle or a piece of wall in the maze, in which the plasmodium cannot go.

1. $Chem^t_{i,j}$ is a floating-point variable. It represents the concentration of chemo-attractants at time t in the area corresponding to the (i, j) cell.
2. $Dir^t_{i,j}$ is a floating-point variable. It indicates the direction of the attraction of the plasmodium bu the chemicals produced by the FS.
3. $Plasm^t_{i,j}$ is a floating-point variable. It indicates the volume of the cytoplasmic material of the plasmodium in the corresponding (i, j) cell.
4. $MinTube^t_{i,j}$ is a one-bit variable. It indicates if the (i, j) cell is included in the final path of tubular network that is formed inside the plasmodium's body.

Furthermore, the results of the CA model are highly affected by some parameters that are defined at the beginning of the modeling process. These parameters are:

– the amount of CA cells that the experimental area is divided to,
– the parameters for the discrete diffusion equation for the cytoplasm of the plasmodium $(pp1, pp2, pp3)$,
– the parameters for the discrete diffusion equation of the chemo-attractants $(cp1, cp2, cp3)$,
– the minimum concentration of chemo-attractants that affect the plasmodium's foraging behavior and,
– the extent that the plasmodium is affected by the chemo-attractants $(0 < Dir < 1)$.

The discrete diffusion equation is used to describe the exploration of the available area by the cytoplasmic material of the plasmodium and the spread of the chemo-attractants produced by FS. The discrete diffusion equation for the plasmodium is given by eq. 9:

$$\begin{aligned} Plasm^{t+1}_{i,j} &= Plasm^t_{i,j} \\ &+pp1\left\{\left[\left(1+N^t_{i,j}\right)Plasm^t_{i-1,j} - pp3\times Plasm^t_{i,j}\right]\right. \\ &+\left[\left(1+S^t_{i,j}\right)Plasm^t_{i+1,j} - pp3\times Plasm^t_{i,j}\right] \\ &+\left[\left(1+W^t_{i,j}\right)Plasm^t_{i,j-1} - pp3\times Plasm^t_{i,j}\right] \\ &+\left.\left[\left(1+E^t_{i,j}\right)Plasm^t_{i,j+1} - pp3\times Plasm^t_{i,j}\right]\right\} \\ &+pp2\left\{\left[\left(1+NW^t_{i,j}\right)Plasm^t_{i-1,j-1} - pp3\times Plasm^t_{i,j}\right]\right. \\ &+\left[\left(1+SW^t_{i,j}\right)Plasm^t_{i+1,j-1} - pp3\times Plasm^t_{i,j}\right] \\ &+\left[\left(1+NE^t_{i,j}\right)Plasm^t_{i-1,j+1} - pp3\times Plasm^t_{i,j}\right] \\ &+\left.\left[\left(1+SE^t_{i,j}\right)Plasm^t_{i+1,j+1} - pp3\times Plasm^t_{i,j}\right]\right\} \end{aligned} \quad (9)$$

.

The variables $N^t_{i,j}$, $S^t_{i,j}$, $W^t_{i,j}$, $E^t_{i,j}$, $NW^t_{i,j}$, $SW^t_{i,j}$, $NE^t_{i,j}$, $SE^t_{i,j}$ correspond to north, south, west, east, north-west, south-west, north-east, south-east directions, respectively and represent the attraction of the plasmodium by the chemo-attractants to a specific direction. In particular, these variables correspond to $Dir^t_{i,j}$ of CA cell state. If the area around a corresponding CA cell has no chemo-attractants, then the foraging strategy of the plasmodium is uniform and these parameters are equal to zero. In the case that there is higher concentration of chemo-attractants in the cell at direction x from the one in direction y, then $Dir^t_{i,j}$ corresponding to direction x is positive and the $Dir^t_{i,j}$ corresponding to direction y is negative, in order to simulate the non-uniform foraging behavior of the plasmodium.

For the expansion of the chemo-attractants, we make use of the discrete diffusion equation as follows, in each time step.

$$\begin{aligned} Chem^{t+1}_{i,j} &= \left\{Chem^t_{i,j}\right. \\ &+cp1\left[\left(Chem^t_{i-1,j} - cp3\times Chem^t_{i,j}\right)\right. \\ &+\left(Chem^t_{i+1,j} - cp3\times Chem^t_{i,j}\right) \\ &+\left(Chem^t_{i,j-1} - cp3\times Chem^t_{i,j}\right) \\ &+\left.\left(Chem^t_{i,j+1} - cp3\times Chem^t_{i,j}\right)\right] \\ &+cp2\left[\left(Chem^t_{i-1,j-1} - cp3\times Chem^t_{i,j}\right)\right. \\ &+\left(Chem^t_{i+1,j-1} - cp3\times Chem^t_{i,j}\right) \\ &+\left(Chem^t_{i-1,j+1} - cp3\times Chem^t_{i,j}\right) \\ &+\left.\left.\left(Chem^t_{i+1,j+1} - cp3\times Chem^t_{i,j}\right)\right]\right\} \end{aligned} \quad (10)$$

.

The variables $Chem^t_{i-1,j}$, $Chem^t_{i+1,j}$,$Chem^t_{i,j-1}$, $Chem^t_{i,j+1}$, $Chem^t_{i-1,j-1}$, $Chem^t_{i+1,j-1}$, $Chem^t_{i-1,j+1}$, $Chem^t_{i+1,j+1}$ represent the concentration of the chemo-attractants of the north, south, west, east,north-west,south-west,north-east and south-east neighbor of the central CA cell (i, j), respectively.

In regards to the provided simulations, we firstly initialize the parameters. We set one specific spot in the maze as the beginning mass value of the plasmodium with a really high value, for example $Plasm^t_{i,j} = 30000$. In another cell, we set the food spot which has also a very big initial value, i.e. $Chem^t_{i,j} = 30000$. Simultaneously, for these two specific CA cells variable $MinTube^t_{i,j}$ is set equal to 1. The parameters for the discrete diffusion equations are declared in Table 1.

Table 1: Parameter values for the discrete diffusion equations.

cp1	cp2	cp3	pp1	pp2	pp3
0.05	0	1	0.05	0	1

Then, an iterative execution of the diffusion equations gives the values of $Plasm^t_{i,j}$ and $Chem^t_{i,j}$ for all the cells in the CA grid. After a few time steps, the procedure stops and the algorithm designs the minimum tubular network based on the values of the $Plasm^t_{i,j}$ variable. When a cell's $MinTube^t_{i,j} = 1$ then the algorithm searches which of its neighbors has the greater value of $Plasm^t_{i,j}$. When it finds it, the $MinTube^{t+1}_{i,j}$ value of this neighbor changes from 0 to 1. This procedure is repeated until the final, minimum tube is created between the cell that the plasmodium was first introduced to the cell with the FS.

6 GPU implementation

In the past few years, the GPUs have become more popular of being effective at manipulating computer graphics. In addition, the architecture of GPUs provide highly parallel structure that makes them more effective than general-purpose CPUs for a wide range of complex algorithms. The term GPGPU (General Purpose computing on Graphics Processing Units) refers to the use of the GPU processor as a parallel device for purposes other than graphic elaboration. NVIDIA released an advanced programming model for its own line of GPUs, the Compute Unified Device Architecture (CUDA). CUDA was created for developing applications for this kind of platforms and it was the main reason for the great success and enormous spread of the GPGPU applications. Fig. 7 represents a generic model of the CUDA architecture, where the system consists of a host that is a traditional CPU and one or more compute devices (GPUs) that are massively data-parallel coprocessors. Host and GPU are connected and communicate via a PCI Express bus. Each GPU device processor supports the Single-Program Multiple Data (SPMD) model. Each GPU has a number of Streaming Multiprocessors (SM), while each SM has eight parallel thread processors called Streaming Processors (SP).

The main reason to work with CUDA is to execute the data-parallel and compute-intensive portions of applications on GPU instead of on classic CPU. For this purpose we use kernels which are functions callable from the host and executed in parallel for each thread on a GPU as presented in fig. 8. For every kernel, GPU is configured with a number of threads and blocks of them. The grouping of block is called grid. All the threads in a grid execute the same kernel functions.

On a GPU there are several levels of memory. The communication between the CPU host and the GPU device is performed through global memory. This memory can deliver higher memory bandwidth than the traditional CPU memory. It is measured that is about 20 times more efficient to access the global memory of the GPU than the CPU memory. But this memory is not cached and there also other memories locally to the multiprocessors that can be used. There is the option to

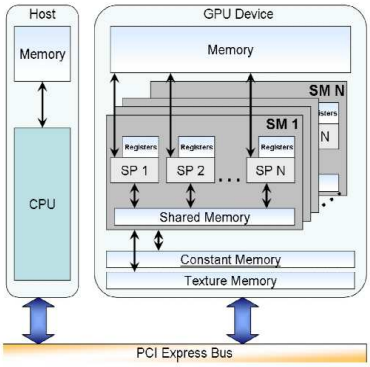

Fig. 7: Standart Architecture of a computer included CPU and GPU. [73]

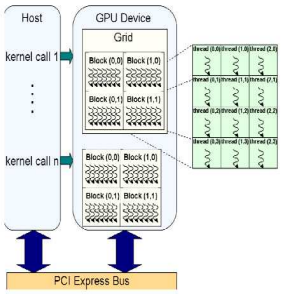

Fig. 8: Descripton of CUDA architecture [73].

use the shared memory of the GPU, which is a fast memory located on the multiprocessors and shared by threads of each thread block. This memory area provides a way for threads to communicate within the same block. There are also registers among streaming multiprocessors that are partitioned among the threads running on them and they constitute faster memory access. There are also some other memories mainly used for graphic operation such as constant or texture memory.

The reason why GPGPU programming is used in CA models can be explained easily when referring to the CA's parallel nature [74]. First, the CA grid seems very similar with the grid of the global memory of the GPU. CUDA provides the ability to assign every CA cell in every thread and make the executions in a synchronous and fully parallel way. The local interaction of the neighbors that CA theory proposes is another very important feature that makes these implementations very suitable and extremely fast thus increasing their performance. The basic idea when computing a CA model in GPU, which is also used in our implementation, is the following:

– First we use two memory regions to store the data processed. More specifically we use one region for the $CA_{current}$, which indicates the CA substates before the

calculations and one other region for the CA_{next} which indicates the CA substates after the calculations.
– Using the $CA_{current}$ we compute the next state of all the CA cells in parallel.
– Finally, the switching procedure occurs between the $CA_{current}$ and the CA_{next} in each time step.

In this implementation, the initial CA data are stored to the global memory of the device as other CA implementations also do [43]. The main steps of our algorithm are:

1. Split the CA substates into different grids. First, we create a 2-dimensional grid for the calculation of the expansion of chemo-attractants, one 2-dimensional grid for the calculation of the mass of the the plasmodium, one 2-dimensional grid which holds the type of the CA cells and finally a 2-dimensional grid for the calculation of the tubes.
2. Then, we assign a kernel to process and make the necessary calculations for every grid we mentioned before. More specifically, we assign a kernel to hold and update the type of the CA cell, one kernel for the computation of the discrete diffusion equation of the chemo-attractants, one kernel for the calculation of the discrete diffusion equation of mass of the plasmodium and one kernel for the formation of the final tubular network with the minimum distance.
3. An initialization of the current state for all the kernels happens through a CPU-GPU memory copy operation (i.e. from host to device global memory).
4. The appropriate kernel run in each time step and makes the calculations by using the information of the current substate of the CA state.
5. At the end of each time step, we assign another kernels to make the device to device memory copy operation. This procedure updates the $CA_{current}$ substate with the CA_{next} substate.
6. When the simulation is completed, the final state of the CA is being retrieved from the global memory of the device to present the results.

For the proposed GPU implementation of the plasmodium CA model we used the graphics card NVIDIA GT640. The CPU we used is the Intel 3770k with a Kingston 16GB RAM. The system runs on Windows 8, with CUDA-C programming language and with the help of Microsoft Visual Studio 2012. In fig. 9 we can see the virtual maze we took into account as the majority of the proposed models mentioned before [1].

In correspondence a CA grid of 50×50 cells was implemented to form the depicted maze. In fig. 10 the first steps of the implementation are presented. We can see the oat flakes in the two different sites of the maze. The left yellow flake is the starting point of the plasmodium. The right flake is the target FS that the plasmodium tries to find by exploring the maze in one pass. The yellow tubes represent the concentration of the plasmodium's mass while is starting to navigate through the maze uniformly.

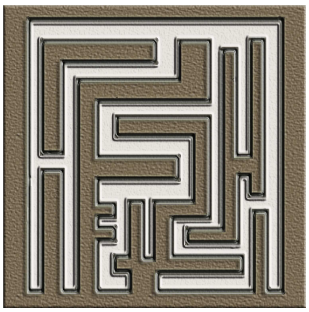

Fig. 9: The maze used in the implementation adopted by [1].

The blue tubes, are the chemo-attractants released by FS and traveling through the maze also uniformly.

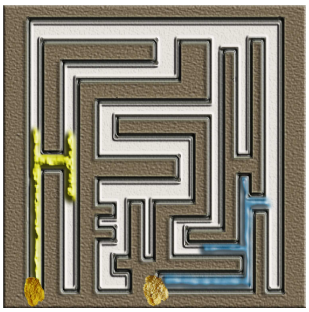

Fig. 10: Representation of the first time steps of the algorithm.

In fig. 11 the greater expansion of the plasmodium's mass and the expansion of the chemo-attractants is presented. Until this time step, the two different tubular networks spread uniformly. While the chemo-attractants spread, their quantity is reduced proportional to the distance. So the path with the minimum distance is going to have the greater quantity of the chemo-attractants and is going to have greater attraction to the plasmodium. From this point and for the next time steps, the plasmodium meets the chemo-attractants of the FS for the first time. More specifically, the CA sub-state $Dir^t_{i,j}$ is starting to take negative or positive values, so the plasmodium can determine the path with the strongest direction of the chemo-attractants. The discrete diffusion equation of the plasmodium's mass is not uniform anymore for those CA cells.

In fig. 12 the algorithm has moved a few time steps forward. It is obvious that the plasmodium chooses the strongest attraction and follows this path. Simultaneously, the plasmodium tubes on other sites are significantly reduced.

Finally, fig. 13 presents the final result of the experiment. The plasmodium reached the FS via the

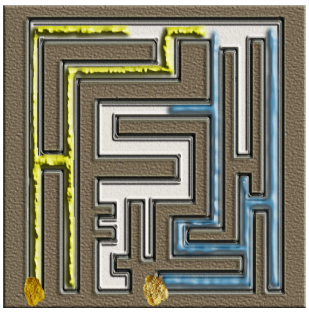

Fig. 11: Representation of the time when the plasmodium meets the chemo-attractants of the FS for the first time .

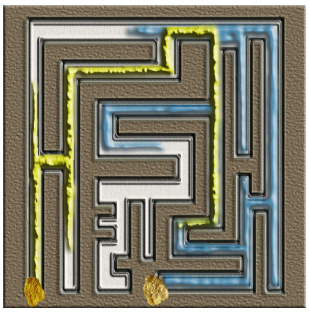

Fig. 12: The plasmodium follows the strongest attraction of the chemo-attractants.

chemo-attractants and by following the the strongest attraction managed to choose and create the minimum path between its source and the target FS. So, the simulation of the proposed CA model to CUDA programming model was successful. In [15] the time needed for the serial software implementation in MATLAB was approximately 45 seconds. The time needed for the presented solution with CUDA is only 2.47 seconds. Therefore, the increase in the performance in our parallel implementation is about 18.2 times more than the serial one. In this point, it is worth mentioning that maybe a serial implementation using a model in VS2012 would produce a more fair result against CUDA. Nevertheless, it should also be mentioned that the graphics card we used is not the spearhead of NVIDIA's devices. NVIDIA GT640 (600 series) is somehow old and is constructed according to Kepler architecture. Nowadays graphics cards (900 series) make also use of Maxwell architecture and have improved capabilities in calculations, in programming and simultaneously in energy efficiency. Therefore, the use of such a device could lead to much greater performance and acceleration of the presented simulations.

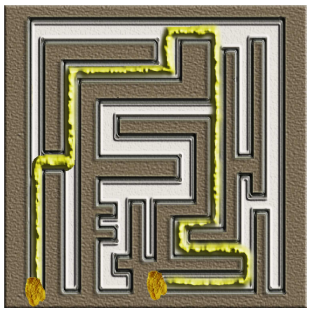

Fig. 13: The final result of the simulation.

7 Conclusions

In this paper, we proposed a parallel General Purpose computing on Graphics Processing Units (GPGPU) implementation of a CA model in order to describe as effectively as previous implementations found in literature, the behavior of *Physarum polycephalum* in a maze and in order to increase its performance. It is shown that this parallel computing approach takes advantage of the inherit parallelism of CA and outperforms the classic serial ones. As a result, we have an accelerated virtual parallel laboratory for *Physarum polycephalum*. For future work, it would be very interesting to search how much acceleration can the GPU achieve if other types of memory are used. For example, if an implementation will handle only the active cells in each time step, and assign them to the shared memory of the GPU then new increment of performance can be probably achieved. The application of the proposed GPGPU based CA *Physarum* model to other hard complex problems, such as Traveling Salesman Problem and other path-planning problems, would be also part of our future work.

References

[1] T. Nakagaki, H. Yamada, Á. Tóth, Intelligence: Maze-solving by an amoeboid organism, Nature, **407:6803** 470-470 (2000).
[2] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, 2004.
[3] A. Adamatzky, Physarum machines: Computers from slime mould, Volume 74. World Scientific, 2010.
[4] J. Jones, A morphological adaptation approach to path planning inspired by slime mould, International Journal of General Systems 1-13 (2014).
[5] J. Jones, A. Adamatzky, Computation of the travelling salesman problem by a shrinking blob, Natural Computing **13:1** 1-16 (2014).
[6] A. Adamatzky, Slime mold solves maze in one pass, assisted by gradient of chemo-attractants, IEEE Transactions on NanoBioscience, **11:2**, 131-134 (2012).
[7] A. Adamatzky, Growing spanning trees in plasmodium machines, Kybernetes **37:2**, 258-264 (2008).

[8] A. Adamatzky (Ed.) Bio-Evaluation of World Transport Networks, World Scientific, 2012.
[9] S. Tsuda, M. Aono, Y.-P. Gunji, Robust and emergent physarum logical-computing, Biosystems, **73:1** 45-55, (2004).
[10] A. Schumann, A. Adamatzky, Physarum spatial logic, New Mathematics and Natural Computation, **7** 483-498 (2011).
[11] A. Adamatzky, T. Schubert, Slime mould microfluidic logical gates, Materials Today, **17:2** 86-91 (2014).
[12] A. Schumann, K. Pancerz, J. Jones, Towards Logic Circuits Based on Physarum Polycephalum Machines: the Ladder Diagram Approach, In: A. Cliquet Jr, G. Plantier, T. Schultz, A. Fred, H. Gamboa (Eds.), Proceedings of the 7th International Conference on Biomedical Electronics and Devices (BIODEVICES2014), Angers, France, March 3-6, (2014)
[13] A. Ishiguro, M. Shimizu, T. Kawakatsu, A modular robot that exhibits amoebic locomotion, Robot. Auton. Syst., **54** 641-50, (2006).
[14] S. Tsuda, K. Zauner, Y. Gunji, Robot control with biological cells, Biosystems, **87**, 215-23 (2007).
[15] M.-A. I. Tsompanas and G. Ch. Sirakoulis, Modeling and hardware implementation of an amoeba-like cellular automaton, Bioinspiration & Biomimetics **7:3** 036013 (2012).
[16] Y. P. Gunji, T. Shirakawa, T. Niizato and T. Haruna, Minimal model of a cell connecting amoebic motion and adaptive transport networks, *Journal of theoretical biology* **253(4)** (2008) 659–667.
[17] Y. P. Gunji, T. Shirakawa, T. Niizato, M. Yamachiyo, I. Tani, An adaptive and robust biological network based on the vacant-particle transportation model, Journal of Theoretical Biology, **272(1)** 187-200 (2011).
[18] A. Tero, R. Kobayashi and T. Nakagaki, Physarum solver: A biologically inspired method of road-network navigation. *Physica A* **363(1)** (2006) 115–119.
[19] R. Kobayashi, A. Tero and T. Nakagaki, Mathematical model for rhythmic protoplasmic movement in the true slime mold, *J. Math. Biol*. **53** (2006) 273-86.
[20] T. Nakagaki, H. Yamada and M. Ito, Reaction diffusion advection model for pattern formation of rhythmic contraction in a giant amoeboid cell of the Physarum plasmodium *J. Theor. Biol*. **197** (1999) 497-506.
[21] A. Adamatzky,If BZ medium did spanning trees these would be the same trees as Physarum built, *Phys. Lett. A* **373** (2009) 952-956.
[22] J. Jones, Approximating the behaviours of Physarum polycephalum for the construction and minimisation of synthetic transport networks, *In Unconventional computation* (Springer Berlin Heidelberg, 2009),191–208.
[23] V. S. Kalogeiton, D. P. Papadopoulos, G. Ch. Sirakoulis, Hey Physarum! Can you Perform SLAM?, International Journal of Unconventional Computing **10:4** 271-293 (2014).
[24] T. Shirakawa, S. Hiroshi, I. Shinji, Construction of living cellular automata using the Physarum plasmodium, International Journal of General Systems **44(3)** 292-304 (2015).
[25] M.-A. I. Tsompanas, G. Ch. Sirakoulis, A. Adamatzky, Evolving Transport Networks with Cellular Automata Models Inspired by Slime Mould, accepted for publication in IEEE Transactions on Cybernetics (2015).
[26] J. Von Neumann, Theory of self-reproducing automata, University of Illinois Press Champaign, IL, USA, 1966.
[27] K. Zuse, Rechnender Raum (Calculating Space), (1969).
[28] G. Ch. Sirakoulis and S. Bandini, (Eds.) Cellular Automata - 10th International Conference on Cellular Automata for Research and Industry, ACRI 2012, Proceedings, Lecture Notes in Computer Science, **7495** Springer (2012).
[29] J. Was, G. Ch. Sirakoulis, S. Bandini, (Eds.) Cellular Automata - 11th International Conference on Cellular Automata for Research and Industry, ACRI 2014, Proceedings, Lecture Notes in Computer Science, **8751** Springer (2014).
[30] G. Ch. Sirakoulis, A. Adamatzky, Robots and Lattice Automata, Springer, 2015.
[31] P. Rosin, A. Adamatzky, Xianfang Sun, (Eds.) Cellular Automata in Image Processing and Geometry, Springer, 2014.
[32] G. Ch. Sirakoulis, I. Karafyllidis, D. Soudris, N. Georgoulas, A. Thanailakis, A new simulator for the oxidation process in integrated circuit fabrication based on Cellular Automata, Modelling and Simulation in Materials Science and Engineering, **7:4**, 631-640 (1999).
[33] G. Ch. Sirakoulis, I. Karafyllidis, A. Thanailakis, A Cellular Automaton Methodology for the Simulation of Integrated Circuit Fabrication Processes, Future Generation Computer Systems, **18:5**, 639-657 (2002).
[34] V. Mardiris, G. Ch. Sirakoulis, Ch. Mizas, I. Karafyllidis, A. Thanailakis, A CAD system for modeling and Simulation of Computer Networks using Cellular Automata, IEEE Transactions on Systems, Man and Cybernetics Part C , **38:2**, 253-264 (2008).
[35] Ch. Mizas, G. Ch. Sirakoulis, V. Mardiris, I. Karafyllidis, N. Glykos, R. Sandaltzopoulos, Reconstruction of DNA sequences using Genetic Algorithms and Cellular Automata: towards mutation prediction?, Biosystems, **92:1**, 61-68 (2008).
[36] K. Ioannidis, G. Ch. Sirakoulis, I. Andreadis, A path planning method based on Cellular Automata for Cooperative Robots, Applied Artificial Intelligence, **25:8**, 721-745 (2011).
[37] L. Nalpantidis, G. Ch. Sirakoulis, A. Gasteratos, Non-probabilistic cellular automata-enhanced stereo vision simultaneous localisation and mapping (SLAM), Measurement Science and Technology, **22:11**, 114027 (2011).
[38] I. G. Georgoudas, G. Ch. Sirakoulis, I. Andreadis, An Anticipative Crowd Management System Preventing Clogging in Exits during Pedestrian Evacuation Processes, IEE Systems, **5:1**, 129-141 (2011).
[39] A. Tsiftsis, I. G. Georgoudas, G. Ch. Sirakoulis, Real Data Evaluation of a Crowd Supervising System for Stadium Evacuation and its Hardware Implementation, accepted for publication in IEE Systems.
[40] P. Progias, G. Ch. Sirakoulis, An FPGA Processor for Modelling Wildfire Spread, Mathematical and Computer Modeling, **57:5-6**, 1436-1452 (2013).
[41] G. Ch. Sirakoulis, I. Karafyllidis, A. Thanailakis, V. Mardiris, A methodology for VLSI implementation of Cellular Automata algorithms using VHDL, Advances in Engineering Software, **32:3**, 189-202 (2001).
[42] G. Ch. Sirakoulis, I. Karafyllidis, Cellular Automata and Power Consumption, Journal of Cellular Automata, **7:1** 67-80 (2012).

[43] D. DAmbrosio, Efficient application of GPGPU for lava flow hazard mapping, The Journal of Supercomputing **65:2**, 630-644 (2013).
[44] H. Mróz, J. Ws, P. Topa, The Use of GPGPU in Continuous and Discrete Models of Crowd Dynamics, Parallel Processing and Applied Mathematics, Springer Berlin Heidelberg, 679-688 (2014).
[45] P. Quesada-Barriuso, D. B. Heras, F. Argüello, Efficient 2D and 3D watershed on graphics processing unit: block-asynchronous approaches based on cellular automata, Computers & Electrical Engineering **39:8**, 2638-2655 (2013).
[46] R. S. Campos, M. Lobosco, R. Weber dos Santos, A GPU-based heart simulator with mass-spring systems and cellular automaton, The Journal of Supercomputing **69:1** 1-8 (2014).
[47] F. A. Sousa, R. J. N. Dos Reis, J. C. F. Pereira, Simulation of surface fire fronts using fireLib and GPUs, Environmental Modelling & Software, **38**, 167-177 (2012).
[48] T. Bajzat, E. Hajnal, Cell automaton modelling algorithms: Implementation and testing in GPU systems, Intelligent Engineering Systems (INES), 2011 15th IEEE International Conference on. IEEE, 2011.
[49] C Cuda Programming guide, NVIDIA Corporation, July (2012).
[50] D. B. Kirk, W. H. Wen-mei, Programming massively parallel processors: a hands-on approach, Newnes, 2012.
[51] J. Holland, Adaptation in Natural and Artificial Systems. Cambridge, MA: MIT Press, 1992.
[52] W. McCulloch, W. Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics **5:4**, 115133 (1943).
[53] A. Adamatzky, R. Armstrong, J. Jones, Y. P. Gunji, On creativity of slime mould, International Journal of General Systems **42:5**, 441-457 (2013).
[54] B. DeLacyCostello, A. Adamatzky, Assessing the chemotaxis behavior of Physarum polycephalum to a range of simple volatile organic chemicals, Communicative & integrative biology **6:5** (2013).
[55] T. Nakagaki, H. Yamada, M. Hara, Smart network solutions in an amoeboid organism, Biophysical chemistry **107:1**, 1-5 (2004).
[56] A. Adamatzky, J. Jones, Road planning with slime mould: if Physarum built motorways it would route M6/M74 through Newcastle, International Journal of Bifurcation and Chaos **20:10**, 3065-3084 (2010).
[57] A. Adamatzky, G. J. Martínez, S. V. Chapa-Vergara, R. Asomoza-Palacio, C. R. Stephens, Approximating Mexican highways with slime mould, Natural Computing **10:3** 1195-1214, (2011).
[58] A. Adamatzky, R. Alonso-Sanz, Rebuilding Iberian motorways with slime mould, Biosystems **105:1**, 89-100 (2011).
[59] A. Adamatzky, S. G. Akl, Trans-Canada slimeways: Slime mould imitates the canadian transport network, *IJNCR*, **2:4**, 31-46 (2011).
[60] A. I. Adamatzky, Route 20, Autobahn 7, and Slime Mold: Approximating the Longest Roads in USA and Germany With Slime Mold on 3-D Terrains, IEEE Transactions on Cybernetics, **44:1**, 126-136 (2014).
[61] A. Adamatzky and T. Schubert, Schlauschleimer in reichsautobahnen: Slime mould imitates motorway network in germany," *Kybernetes* **41:7** 1050-1071 (2012).
[62] A. Adamatzky, S. Akl, R. Alonso-Sanz, W. van Dessel, Z. Ibrahim, A. Ilachinski, J. Jones, A. V. D. M. Kayem, G. J. Martinez, P. de Oliveira, M. Prokopenko, T. Schubert, P. Sloot, E. Strano, and Xin-She Yang, Are motorways rational from slime mould's point of view?, Int. J. Parallel Emerg. Distrib. Syst., **28:3** 230-248 (2013).
[63] T. Shirakawa, A. Adamatzky, Y. P. Gunji, Y. Miyake, On simultaneous construction of Voronoi diagram and Delaunay triangulation by Physarum polycephalum, International Journal of Bifurcation and Chaos **19:9**, 3109-3117 (2009).
[64] T. Shirakawa, Y. P. Gunji, Emergence of morphological order in the network formation of Physarum polycephalum, Biophysical chemistry **128:2**, 253-260 (2007).
[65] . Shirakawa, Y. P. Gunji, Y. Miyake, An associative learning experiment using the plasmodium of Physarum polycephalum, Nano Communication Networks **2:2** 99-105 (2011).
[66] T. Shirakawa, H. Sato, Construction of molecular learning network, Journal of Advanced Computational Intelligence and Intelligent Informatics **17** 913-918 (2013).
[67] A. Adamatzky, J. Jones, On using compressibility to detect when slime mould completed computation, Complexity (2015).
[68] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, T. Nakagaki, Rules for biologically inspired adaptive network design, Science, **327:5964** 439-442 (2010).
[69] A. Tero, R. Kobayashi, T. Nakagaki, A mathematical model for adaptive transport network in path finding by true slime mold, Journal of theoretical biology **244:4** 553-564 (2007).
[70] A.Tero, K. Yumiki, R. Kobayashi, T. Saigusa, T. Nakagaki, Flow-network adaptation in Physarum amoebae, Theory in Biosciences **127:2** 89-94 (2008).
[71] B. Chopard, M. Droz, Cellular automata **24**, Cambridge: Cambridge University Press, 1998.
[72] M.-A. I. Tsompanas, G. Ch. Sirakoulis, A. I. Adamatzky, Physarum in silicon: the Greek motorways study, Natural Computing 1-17 (2014).
[73] P. Vidal, E. Alba, A multi-GPU implementation of a Cellular Genetic Algorithm, Evolutionary Computation (CEC), IEEE Congress on. IEEE, 1-7 (2010).
[74] N. Dourvas, G. Ch. Sirakoulis, Ph. Tsalides, GPU Implementation of Physarum Cellular Automata Model, Proceedings of Symposium: 1st International Symposium on Artificial, Biological and Bio-Inspired Intelligence (ABBII) organized within the 12th International Conference on Numerical Analysis and Applied Mathematics (ICNAAM 2014), Rhodes, September 2014.

Nikolaos I. Dourvas Nikolaos I. Dourvas was born in Thessaloniki in 1990. He received his Diploma degree in Electrical and Computer Engineering in 2013 from the Democritus University of Thrace (DUTh) in Greece. He was in the organizing comittee of the 5th Panhellenic Electrical and Computer Engineering Students Conference (SFHHMY 5) in early April 2012 in Xanthi, Greece. He is now working on his M.Sc degree at the same university. He is member of IEEE and member of the Robotics Team in Democritus University of Thrace. His main interests are cellular automata, modeling of large scale systems, parallel programming, hardware design and bio-inspired algorithms.

Georgios Ch. Sirakoulis Dr. Georgios Ch. Sirakoulis received his Diploma degree in Electrical and Computer Engineering (1996) from the Democritus University of Thrace (DUTh), Greece, and for his Diploma Thesis he received a prize of distinction from the Technical Chamber of Greece (TEE). In 2001, he received his PhD in Electrical and Computer Engineering from the Democritus University of Thrace, Greece. He is an Associate Professor with tenure in the Department of Electrical and Computer Engineering, Democritus University of Thrace (2008-today). Moreover, he is EUROPRACTICE representative for DUTh, and he has served as a member at the EU IDEAS programme. Furthermore, he is Associate Editor in "Microelectronics Journal", "Journal of Applied Matehematics", "Recent Patents on Electrical & Electronic Engineering" and "Conference Papers in Computer Science". He is a member of the Institute of Electrical and Electronic Engineering (IEEE), of the IEEE Computer Society, of the Institute of Electrical Engineering (IEE), of the Association of Computing Machinery (ACM), of the Institution of Engineering and Technology (IET), of the European Geosciences Union (EGU), of the International Society for Computational Biology (ISCB), and a member of the TEE. He was also founding member and Vice President of the IEEE Student Branch of Thrace for the period 2000-2001. In 2012, he was the general co-chair of 10th edition of ACRI 2012 Conference (Cellular Automata for Research and Industry) in late September in Santorini, Greece and general co-chair of Panhellenic Electrical and Computer Engineering Students Conference (SFHHMY 5) in early April 2012 in Xanthi, Greece. His courses lab activities are sponsored by ARM, Freescale, Xilinx and Altera.

Philippos Tsalides Philippos G. Tsalides was born in Mirina Limnou, Greece, on October 14th 1953. He received the Diploma degree in electronic engineering from the University of Padova, Padova, Italy, in 1979, and the Ph.D. degree in electrical engineering from the Democritus University of Thrace, Xanthi, Greece, in 1985. He is a Professor of Applied Electronics with the Department of Electrical and Computer Engineering, Democritus University of Thrace. His current research interests include VLSI architectures, VLSI systems, BIST Techniques, LANs, WANs, applications of cellular automata in image processing, as well as in computational systems. He has published a number of papers and three textbooks on VLSI systems, microprocessors, and automated electronic measurements.