GPU Implementation of *Physarum* Cellular Automata 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

Abstract. In the past few decades, there is an increasing number of publications which show that solutions to complex mathematical problems can be found by applying unconventional computing methods. Among other examples, the plasmodium of *Physarum Polycephalum* has been intensively used for solving shortest path(s) problem, various graph problems, evaluation of transport networks, robotic control and many other engineering applications. In this paper we coupled the computing abilities of slime mould with one of the most powerful parallel computational models, namely Cellular Automata (CAs). CAs can capture the essential features of systems which global behavior emerges from the collective effect of simple components, which interact locally. Moreover, a Graphical Processing Unit (GPU) implementation will exploit the prominent feature of parallelism that CA structures inherently possess in contrast to the serial computers, thus accelerating the response of the proposed model. As a result, a slime mould CA based model in graphics processing unit (GPU) using CUDA programming model to describe and mimic the behavior of a plasmodium in a maze. In this way we are able to produce a virtual lab speeding up significantly the biological paradigm in GPU.

Keywords: Cellular automata, *Physarum Polycephalum*, Bio-inspired algorithm, Parallel Computing
PACS: 89.20.Ff,89.90.+n,80

INTRODUCTION

Biologically inspired algorithms or bio-inspired algorithms constitute a type of algorithms that imitate specific phenomena from nature. In our study, we will try to provide a accelerating model to mimic the behavior of plasmodium of *Physarum Polycephalum*. The plasmodium of this slime mould is a large amoeba-like cell consisting of a dendritic network of tube-like structures (pseudopodia). It changes its shape as it crawls over a plain agar gel and, if food is placed at two different points, it will put out those pseudopodia that connect the two food sources (FSs). Recent publications demonstrate that this simple organism has the ability to find the minimum-length solution between two points in a labyrinth [1] showing a sample of intelligence. Subsequent research has moved slime mould computational abilities to new areas such as spatial representations of various graph problems[2], logical machines [3], Simultaneous Localization and Mapping (SLAM) [4], etc. So far, there is a variety of modeling approaches which also are implemented by a variety of tools.The bibliography presents some purely spatial Cellular Automata (CAs) models [5], [6], a two-variable Oregonator model of Belousov-Zhabotinsky (BZ) medium [7], etc.

CAs are models of physical systems, where space and time are discrete and interactions are local. Prior and more recent works proved that CAs are very effective in simulating physical systems and solving scientific problems, because they can capture essential features of systems where global behavior arised from the collective effect of simple components, which interact locally. In the last two decades, CA has been widely used in many applications like modeling and simulation of physical systems and processes [8] biological modeling involving models for self-reproduction, biological structures and DNA sequences [9], etc. Furthermore, the CAs approach is consistent with the modern notion of unified spacetime. In computer science, space corresponds to memory and time to processing unit. In CAs, memory (CA cell state) and processing unit (CA local rule) are inseparably related to a CA cell [10].

In this paper, a proposed CA model, which describes the behavior of *Physarum Polycephalum*, is implemented in GPU using CUDA programming model in order to accelerate the responding times. It is important to mention that until now there is no single model that completely encapsulates *Physarum's* behavior and we consider only the plasmodium stage of its life. 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, the movement, the internal oscillations or the network adaptation. In this paper we are modeling the movement of the plasmodium from one place in a maze when there is food in another spot of the maze in order to find the minimum path between these two places. The biological experiments show that the plasmodium spreads its pseudopodia trying to reach the food. Simultaneously, the food releases the chemo-attractants

Proceedings of the International Conference on Numerical Analysis and Applied Mathematics 2014 (ICNAAM-2014)
AIP Conf. Proc. 1648, 580019-1–580019-4; doi: 10.1063/1.4912827

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 maze in one pass because it is assisted by a gradient of chemo-attractants propagating from the target food [11]. First, we describe the *Physarum* CA model. Afterwards, the proposed GPU implementation and performance results of the CA model are reported. Finally, conclusions are drawn at the last section.

PHYSARUM CA MODEL

In this section, the CA model designed in [5] to describe the behavior of P.Polycephalum is described. We consider the biological experiment where the plasmodium was starved and then introduced into a specific place in the maze. Moreover, a FS which produces chemo-attractants is placed in another place of the maze. In order to simulate this biological experiment, the area is divided into a matrix of squares with identical areas and each square of the surface is represented by a CA cell. The type of neighborhood that was used in this CA model is the Moore neighborhood which means that we use the north, south, east, west, north-east,north-west, south-east and south-west neighbors.The state of the (i, j) cell at time t, defined as $C^t_{i,j}$ is equal to:

$$C^t_{i,j} = \{Flag_{i,j}, Mass^t_{i,j}, Food^t_{i,j}, DA^t_{i,j}, Tube^t_{i,j}\} \tag{1}$$

- $Flag_{i,j}$ is a variable which indicates the type of area of the corresponding (i, j) cell. The possible values of this variable are 0,1,2,3 and indicate a free area, the spot of the initially placed FS, the spot of the initially placed plasmodium and the spot which represents a wall of the maze respectively.
- $Food^t_{i,j}$ represents the concentration of chemo-attractants at time t in the area corresponding to the (i, j) cell. In order to calculate this variable for every cell, we make use of the diffusion equation of the chemo-attractants in each time step:

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

  The variables $Food^t_{i-1,j}$, $Food^t_{i+1,j}$, $Food^t_{i,j-1}$, $Food^t_{i,j+1}$ represent the concentration of the chemo-attractants of the north,south, west and east neighbor of the central cell (i, j), respectively. The variables $fp1$, $fp2$, $fp3$ are the parameters of the diffusion equation (2) which are chosen after many simulations and have the values 0.05, 0 and 1, respectively.
- $DA^t_{i,j}$ is a variable that indicates the direction of the attraction of the plasmodium by the chemicals produced by the FS. For example, if the area around a corresponding cell has no chemo-attractants, the foraging strategy of the plasmodium is uniform and, thus, these parameters are equal to zero. If there is higher concentration of chemo-attractants in the cell at direction x from the one in direction y, then the parameter corresponding to direction x is positive and the parameter corresponding in the direction y is negative. This happens, in order to more accurately simulate the non-uniform foraging behavior of the plasmodium.
- $Mass^t_{i,j}$ indicates the volume of the cytoplasmic material of the plasmodium in the corresponding (i, j) cell. In order to calculate this variable for every cell, we make use of the diffusion equation of the plasmodium in each time step is given by:

$$\begin{aligned} Mass^{t+1}_{i,j} = & Mass^t_{i,j} + op1 \left\{ [(1 + N^t_{i,j}) Mass^t_{i-1,j} - op3 \times Mass^t_{i,j}] + [(1 + S^t_{i,j}) Mass^t_{i+1,j} - op3 \times Mass^t_{i,j}] \right. \\ & \left. + [(1 + W^t_{i,j}) Mass^t_{i,j-1} - op3 \times Mass^t_{i,j}] + [(1 + E^t_{i,j}) Mass^t_{i,j+1} - op3 \times Mass^t_{i,j}] \right\} \\ & + op2 \left\{ [(1 + NW^t_{i,j}) Mass^t_{i-1,j-1} - op3 \times Mass^t_{i,j}] + [(1 + SW^t_{i,j}) Mass^t_{i+1,j-1} - op3 \times Mass^t_{i,j}] \right. \\ & \left. + [(1 + NE^t_{i,j}) Mass^t_{i-1,j+1} - op3 \times Mass^t_{i,j}] + [(1 + SE^t_{i,j}) Mass^t_{i+1,j+1} - op3 \times Mass^t_{i,j}] \right\} \end{aligned} \tag{3}$$

where $N^t_{i,j}$, $S^t_{i,j}$, $W^t_{i,j}$, $E^t_{i,j}$, $NW^t_{i,j}$, $SW^t_{i,j}$, $NE^t_{i,j}$, $SE^t_{i,j}$ correspond to north, south, west, east, north-west, south-west, north-east, south-east directions. The variables $op1$, $op2$, $op3$ are the parameters of the diffusion equation (3) which are chosen once again after many simulations and have the values 0.05, 0 and 1, respectively.

- Finally, $Tube^t_{i,j}$ is a variable which can take values [0,1] and illustrates if the (i, j) cell is included in the final path of tubular network that is formed inside the plasmodium's body. This tubular network forms the shortest path between the FS and the cell from where the plasmodium started to expand and it is our final solution to the maze problem.

GPU IMPLEMENTATION

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. The reason of the great success and enormous spread of the GPGPU application in the past few years, is CUDA programming model. The basic structure of CUDA is that it provides three key abstractions, namely the hierarchy with which the threads are organized, the memory organization and the functions that execute in parallel, called kernels.

Threads can access different memory locations during execution. Generally, there are three types of memory used in CUDA applications, namely (*a*) the private memory, which is the memory its thread has for its own, (*b*) the shared memory, which is the memory being visible to all threads in a block and (c) the global memory, which is a larger memory on the device board but it is outside the computing chip. In this study, we make use of the global memory of the device. This memory is slower if compared with the shared memory but it can deliver a significantly 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.

The reason why GPGPU programming is used for CAs models can be explained easily when referring to the CAs' parallel nature. The local interaction of the neighbors that CAs methods propose is another fact that makes therse implementations very suitable and very fast. These features make the CAs models ideal to be implemented in parallel computers. The basic idea when computing a CA model in GPU, which is also used in our implementation, can be described as follows: First, we compute the next state of all the cells in parallel. Afterward we use two memory regions to store the data. More specifically, we use one region for the $CA_{current}$, which indicates the CA states before the calculations and one for the CA_{next}, which in turns indicates the CA states after the calculations. Finally, the switching between the $CA_{current}$ and the CA_{next} in each time step takes place.

For this paper we store the CA data to the global memory of the device as other CA implementations do [12]. The steps of the algorithm are: (*a*) Split the CA states and make use of a kernel for every one of them. In more detail, we make use of a kernel to hold the $flag_{i,j}$, one kernel for the computation of the diffusion equation of the chemo-attractants, $food_{i,j}$, and their direction, $DA_{i,j}$, one kernel for the computation of the diffusion equation of the mass of plasmodium, $mass_{i,j}$ and finally one kernel that computes the $Tube_{i,j}$ to find the shortest path in the maze. (*b*) An initialization of the current state for all these kernels happens through a CPU-GPU memory copy operation (i.e. from host to the device global memory). (*c*) Every kernel runs in each time step and makes its calculations by using the information of the states of the other necessary kernels. For example, in order to calculate the $Tube_{i,j}$ we have to know which of the neighbors has the greater mass value. Therefore, we take this information from the kernel that executes the computation of the $Mass_{i,j}$. (*d*) At the end of each CA step, a device to device memory copy operation is used to update the new values in order to continue the calculations in the next CA step. (*e*) When the simulation is completed the final state of the automaton is being retrieved from the global memory of the device to the host through a GPU-CPU memory copy operation.

IMPLEMENTATION RESULTS

For the proposed GPU implementation of the presented slime mould CA based model we used the graphics card NVIDIA GT640. In Figure 1 we can see the simulation result of the experiment. We used a 50×50 CA cells in order to synthesize the maze. Letter 'M' in Figure 1 corresponds to the initial place where the plasmodium is located. Letter 'F' indicates the cell in which the food is located and from where the chemo-attractants are released. The yellow line indicates the plasmodium and the blue one indicates the spreading of the chemo-attractants. In Figure 1.(a) the first steps of the CA model are presented. The plasmodium and the chemo-attractants are spreading in the maze to any

direction. In Figure 1.(b) is presented the time when the plasmodium meets the chemo-attractants. In Figure 1.(c) the plasmodium follows the path with the greater quantity of the chemo-attractants. It is obvious that 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. The same time, the plasmodium reduces its pseudopodia from the other paths. Finally, in Figure 1.(d) the shortest path between the initial mass of the plasmodium and the FS is presented without showing the expansion of the chemo-attractants. The time needed for the presented solution to result is 2.47 seconds. In [5] the time needed for the serial software implementation in MATLAB was approximately 45 seconds. Therefore, the increase in the performance in our implementation is about 18.2 times more than the one in MATLAB.

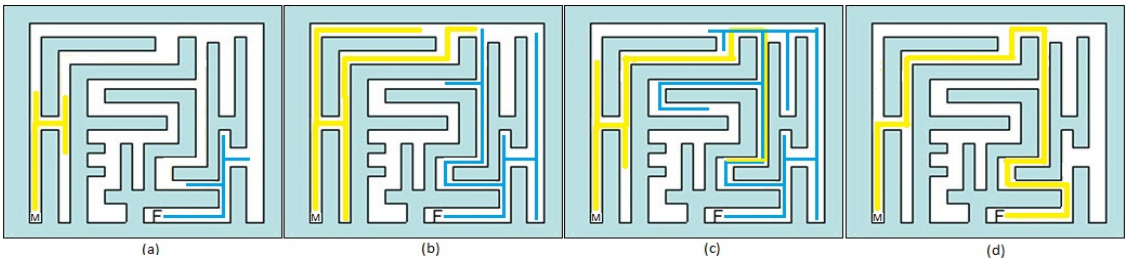

FIGURE 1. Solution to the shortest-path problem in a maze resulting from the GPU implementation of the CA *Physarum* model.

CONCLUSIONS

In this paper we took advantage of the inherit parallelism of CA plasmodium model in order to make a GPU implementation to describe its behavior in a maze. It is shown that this parallel computing approach outperforms the classic serial ones and the model responds faster. As a result we have an accelerated virtual laboratory for *Physarum Polycephalum*. Finally, an aspect of future work is using other approaches for the GPU implementation like the handling only of the cells that are active in order to reduce the bandwidth needed in the copy operations. Furthermore, we could make use of other types of device memory, such as shared memory, to accelerate more the response of our model.

REFERENCES

1. Nakagaki, Toshiyuki, Hiroyasu Yamada, and Agota Toth. "Intelligence: Maze-solving by an amoeboid organism." *Nature* 407.6803 (2000): 470-470.
2. Adamatzky, Andrew. "Growing spanning trees in plasmodium machines." *Kybernetes* 37.2 (2008): 258-264.
3. Adamatzky, Andrew. "*Physarum* machine: implementation of a Kolmogorov-Uspensky machine on a biological substrate." *Parallel Processing Letters* 17.04 (2007): 455-467.
4. Kalogeiton, Vicky, Papadopoulos, Dimitrios, and Georgios Ch. Sirakoulis. "Hey *Physarum*! Can you perform SLAM?" *International Journal of Unconventional Computing*, 10.4 (2014): 271-293.
5. Tsompanas, Michail-Antisthenis I., and Georgios Ch Sirakoulis. "Modeling and hardware implementation of an amoeba-like cellular automaton." *Bioinspiration & Biomimetics* 7.3 (2012): 036013.
6. Gunji, Yukio-Pegio, et al. "An adaptive and robust biological network based on the vacant-particle transportation model." Journal of theoretical biology 272.1 (2011): 187-200.
7. Adamatzky, Andrew. "If BZ medium did spanning trees these would be the same trees as *Physarum* built." *Physics Letters* A 373.10 (2009): 952-956.
8. Sirakoulis, Georgios Ch., and Bandini, Stefania. "Cellular Automata." *Lecture Notes in Computer Science (LNCS)* 7495 (2012).
9. Mizas, Ch, et al. "Reconstruction of DNA sequences using genetic algorithms and cellular automata: Towards mutation prediction?." *Biosystems* 92.1 (2008): 61-68.
10. Sirakoulis, G. Ch, et al. "A methodology for VLSI implementation of cellular automata algorithms using VHDL." *Advances in Engineering Software* 32.3 (2000): 189-202.
11. Adamatzky, Andrew. "Slime mold solves maze in one pass, assisted by gradient of chemo-attractants." *IEEE Transactions on NanoBioscience* 11.2 (2012): 131-134.
12. D'Ambrosio, Donato, et al. "Efficient application of GPGPU for lava flow hazard mapping." *The Journal of Supercomputing* 65.2 (2013): 630-644.