A CUDA-based Implementation of the Cellular Potts Model combined with Lattice-Gas Cellular Automata for Cancer Cell Growth Simulation

Lei Kristoffer R. Lactuan
Institute of Computer Science
University of the Philippines
Los Baños
lk.lactuan@gmail.com

Joseph Anthony C. Hermocilla
Institute of Computer Science
University of the Philippines
Los Baños
jachermocilla@uplb.edu.ph

Mehrdad Ghaemi
Department of Chemistry
Teacher Training University
Tehran, Iran
ghaemi@tmu.ac.ir

Amene Shahrokhi
District Health Center Shahre Ray
Vice-Chancellor for Health
Tehran University of Medical Science
Tehran, Iran
amene_shahrokhi@hotmail.com

ABSTRACT

The process in which cancer cells invade or grow is very complex. Thus researchers rely on simulation to study and understand this phenomenon. Cellular Pots Model (CPM) and the Lattice-Gas Cellular Automata(LGCA) is one model for cancer cell growth simulation. In this work, we present a Compute Unified Device Architecure(CUDA)-based implementation of CPM combined with LGCA and compare its performance in terms of execution time to a CPU-based implementation.

Keywords

Cellular Potts Model, Lattice-Gas Cellular Automata, cancer, cancer cells, necrotic cells, GPU, CUDA, CPU, matrix, necrosis, apoptosis, mitosis, quiescent, proliferation, model, mathematics, biology

1. INTRODUCTION

Mathematics and Biology seem to have no intersections since both disciplines are thought to be far-fetched from each other. Mathematics focuses on the numerical data while Biology focuses on complex data which is nonnumeric. However, mathematics is a powerful tool in which behavioral patterns of complex biological phenomena can be modeled using mathematical equations. This is mainly due to the expansion of both disciplines and collaborations between them, as well as because of the large amount of data accumulated from each discipline.

Cancer is a complex process since genetic changes within the infected cells occur in a sub cellular level [6]. This results into mutations within the cells that cause continuous abnormal divisions and cause destruction on neighboring cells. This manner of cell mutation and destruction is called *cancer cell growth*. Cancer has been marked as one of the diseases that is very dangerous due to its high mortality rate.

The process in which cancer cells invade or grow within a matrix of cells is complex. This makes fully understanding it, a great challenge for researchers. Since the need for fast evaluation and analysis of such complex phenomena currently arises, it attracted interest among scholars to create models for it, particularly carcer cell growth, for use in simulations.

Simulations are imitations of real objects or phenomena which does not necessarily require the actual object being simulated to be present. It is a useful tool in executing these phenomena in real time even if the phenomena being observed is out of season. Some objects that are being simulated are volcanic erruptions, tsunamis, and other disasters. Animal or insect behavior can also be simulated. Some of these animals or insects are bees, fishes, and more.

Simulations, particularly in disasters, allows one to observe what will happen during a disaster given specific parameters. Thus, we can plan ahead on how to prevent the disaster or plan on how to escape from it. This is also applicable on biological simulations wherein one can observe how diseases spread throughout the body as well as how one might know how to treat the disease.

Through the years, a lot of models concerning cancer growth have been developed, simulated, and implemented; such models show cell-to-cell interactions and competitions. This pa-

per focuses on the Cellular Potts Model (CPM) combined with the Lattice-Gas Cellular Automata (LGCA). A cellular automata is a set of cells in a matrix where in each grid within the matrix, the cells evolve according to the set of rules and formulas specified as every time step progresses. The advantage of using this model compared to other simulation models of cancer growth is that this method only needs three parameters, discussed later in the paper, based on the well known physical ground[1].

The cells in a particular living body can be visualized as a matrix where cancer cell growth may occur. Cancer cell growth is the moving of infected cells throughout the matrix. These cells detach themselves from their respective post and moves in order to infect or invade other cells. Models described the growth of cancer cells and their ability to degrade neighboring cells [5].

2. PROBLEM STATEMENT

In time, the number of cancer cells in the system must be determined in order to identify the growth rate. This is very essential since curing cancer is a race against time. Also, growth of cancer cells is happening simultaneously in a system.

NVidia introduced Compute Unified Device Architecture (CUDA) to be used in simulations especially in problems involving intensive computing and highly parallel computations. CUDA makes use of the Graphics Processing Unit (GPU) to speed up processing. Since the model or the behavior of cancer growth is parallel in itself, CUDA is well suited for simulating its behavior [4]. Cellular automata, CPM combined with LGCA in particular, contains steps that can be executed in parallel, and thus can be implemented in CUDA.

3. METHODOLOGY

The general steps in the study follows the outline presented in Figure 1.

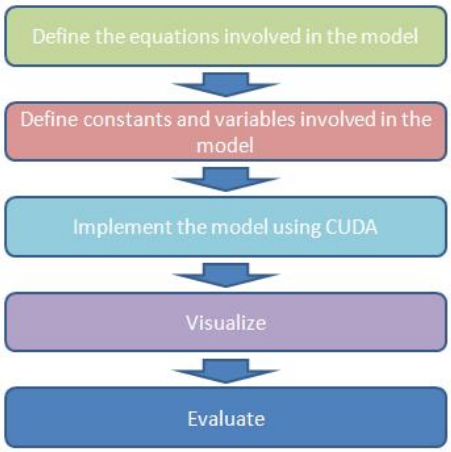

Figure 1: General flowchart used in the study.

CPM and LGCA

A square lattice of size 300x300 is defined wherein each site with coordinates (i, j) is occupied by a cell. Each cell has the ability to *proliferate*, *be quiescent*, or *die* due to *apoptosis* or *necrosis*. Proliferation is the ability of a cancer cell to grow rapid in number; Quiescent is when a cancer cell is inactive or will just stay the same; Death of a cancer cell through apoptosis will replace the location of a cancer cell with a healthy cell, and; Death through necrosis will replace the location of a cancer cell with a necrotic cell. Each cell in the matrix defined is exposed to the same probabilities of whether the cell will proliferate or undergo mitosis(p_m), be quiescent (p_q), or die due to apoptosis (p_a) or necrosis (p_n), the equation for these probabilities are shown below:

$$p_q = \frac{e^{\frac{-E_q}{kT}}}{e^{\frac{-E_q}{kT}} + e^{\frac{-E_m}{kT}} + e^{\frac{-E_a}{kT}} + e^{\frac{-E_n}{kT}}} \quad (1)$$

$$p_m = \frac{e^{\frac{-E_m}{kT}}}{e^{\frac{-E_q}{kT}} + e^{\frac{-E_m}{kT}} + e^{\frac{-E_a}{kT}} + e^{\frac{-E_n}{kT}}} \quad (2)$$

$$p_a = \frac{e^{\frac{-E_a}{kT}}}{e^{\frac{-E_q}{kT}} + e^{\frac{-E_m}{kT}} + e^{\frac{-E_a}{kT}} + e^{\frac{-E_n}{kT}}} \quad (3)$$

$$p_n = \frac{e^{\frac{-E_n}{kT}}}{e^{\frac{-E_q}{kT}} + e^{\frac{-E_m}{kT}} + e^{\frac{-E_a}{kT}} + e^{\frac{-E_n}{kT}}} \quad (4)$$

These probabilities are dependent on the following configuration energies which represent the cell-to-cell interactions:

$$E_q = -\left\{\tfrac{1}{2}(C(C-1)(K_{CC}) + N(N-1)(K_{NN}))\right\} + (C)(N)(K_{NC})$$

$$E_m = -\left\{\tfrac{1}{2}(C(C+1)(K_{CC}) + N(N-1)(K_{NN}))\right\} + (C+1)(N)(K_{NC})$$

$$E_a = -\left\{\tfrac{1}{2}((C-1)(C-2)(K_{CC}) + N(N-1)(K_{NN}))\right\} + (C-1)(N)(K_{NC})$$

$$E_n = -\left\{\tfrac{1}{2}((C-1)(C-2)(K_{CC}) + N(N+1)(K_{NN}))\right\} + (C-1)(N+1)(K_{NC})$$

N in the configuration energy represents the number of necrotic cells while C represents the number of cancer cells at coordinate (i, j). The constants seen here which are K_{CC}, K_{NC}, and K_{NN} are the $cancer - cancer$, $necrotic - cancer$, and $necrotic - necrotic$ connections within the cellular matrix.

Reaction, Propagation, and Redistribution

A 2D array is created for a truth table with 2^5 entries. Each line in the truth table represents each canal combination that might be present in each cell site in the cellular matrix. The 0's in the canal represent the necrotic cells and the 1's represent the cancer cells present in the site. For example, given 00100, the number of cancer cells in this canal is one while the number of necrotic cells is four. A 2D array which holds the reverse of each canal is also needed. In this reversed 2D array, the 1's represent the necrotic cells while the 0's represent the cancer cells. This reversed canal array will be used in the addition process.

The next step is to count the number of cancer cells per canal and store it in an array, this is used in the redistribution step.

Canal	Cancer Cells	Canal	Cancer Cells
00000	0	10000	1
00001	1	10001	2
00010	1	10010	2
00011	2	10011	3
00100	1	10100	2
00101	2	10101	3
00110	2	10110	3
00111	3	10111	4
01000	1	11000	2
01001	2	11001	3
01010	2	11010	3
01011	3	11011	4
01100	2	11100	3
01101	3	11101	4
01110	3	11110	4
01111	4	11111	5

Table 1: Number of cancer cells present in a canal.

For each combination of the number of necrotic and cancerous cells, in assigning different probabilities for mitosis(p_q), apoptosis(p_a), necrosis(p_n), and quiescent(p_q) which are dependent on their respective energies, the number of the cancer cells in the C variable and necrotic cells in the N variable are substituted to the energy formulas. Given the number of necrotic and cancerous cells present, the probabilities that are most likely to happen can be identified. This process is decribed below.

1. Having any number of necrotic cells from zero to five and having zero cancer cells falls to the cluster of being quiescent.

2. Having zero necrotic cells with four or less number of cancer cells or having the sum of the number of necrotic cells and cancer cells less than or equal to four opens the probability where mitosis, apoptosis, necrosis, and quiescent may happen. Probability for mitosis equals p_m. Probability for apoptosis equals $p_a + p_m$ but will later on be just p_a because mitosis cannot happen if the cancer cell bacomes healthy again. Probability for necrosis equals $p_n + p_a + p_m$ but will later on be just $p_n + p_m$ because necrosis and apoptosis cannot happen at the same time in the same cell. Probability for quiescent equals 1 but will later on become $1 - (p_q + p_m + p_a + p_n)$.

3. Having zero necrotic cells with five cancer cells or having the sum of the number of necrotic and cancer cells equal to five completely deletes the probability for healthy cells to reproduce. $p_m = 0$. Probability for apoptosis equals p_a. Probability for necrosis equals $p_n + p_a$ but will later on be just p_n because necrosis and apoptosis cannot happen at the same time in the same cell. Probability for nothing to happen equals 1 but will later on be equal to $1 - (p_q + p_m + p_a + p_n)$. Else, quiescent will take place.

Each cell contains a value of 0 to 31 which holds the combinations of 0's and 1's that represent each canal, which is also the state of the cell. The fate of the cells depends on the neighborhood surrounding it. Each direction of the neighbors is represented by numbers 1, 2, 4, 8, and 16 which is the direction of influence by the neighborhood; 1 on the left, 2 downward, 4 on the right, 8 upward, and 16 on the site of the cell currently being examined. The propagation of the cells is dependent on the movement and the movement may change depending on the collision of cells. They propagate according to the comparison of the movement towards the neighboring cell that inhibits the direction of the movement. Propagation is done by comparing first each neighborhood of the cell site by the corresponding direction of each cell in the neighborhood from the site using a bitwise AND operator and then compare all the neighboring cells using a bitwise OR operator.

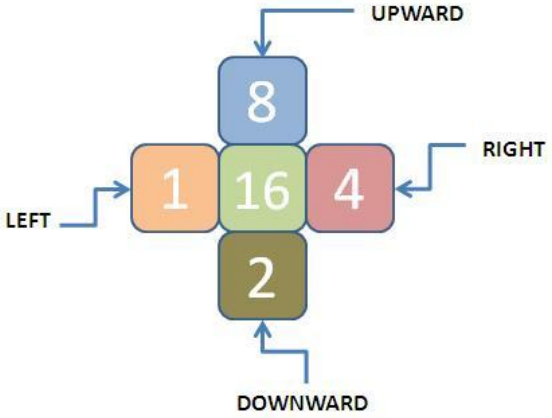

Figure 2: Site movement values.

Redistribution of cells happens first to the necrotic cells since these are less motile than cancer cells. Any cell has a value that represents its state which is represented by binary numbers of base 2^5 which are the canals. In each configuration or canal, count the number of cells. The array where the counting was stored is used.

1. If the count of necrotic cells in the canal of the site is zero, then there are no necrotic cells present. If the

count of cancer cells in the canal of the site is zero then there are no cancer cells present in the site. Else, due to abnormal union of the cancer cells, the site where the cancer cell is present will change its state based on the relative magnitude of chemotoxic materials in the site which is assumed by the authors of the model as equal to the relative number of necrotic cells adjacent to the site.

2. If the count of necrotic cells in the canal of the site is one, then that site will assume a value of sixteen for the necrotic cells since no change will happen to the necrotic cells on the site. The part on the cancer cells is the same as mentioned.

3. If the count of necrotic cells is more than one, then due to abnormal union of the cells, the site where the cell is present will change its state based on the relative magnitude of chemotoxic materials in the site for both necrotic and cancer cells. The exception is when the number of cancer cells present is equal to zero, then the abnormal union of cells will only happen to the necrotic cells.

The action at each time step is to know the final state of a cancer cell, when it will proliferate (mitosis), do apoptosis, be necrotic, or be quiescent, which is dependent on the probabilities obtained using the formulas. If a site is not occupied by any cancer cell, then nothing happens on the site. Else, a random number is generated and compared to the probabilities, given the number of necrotic and cancer cells in the site. When the random value:

1. is in the range $[0, p_m)$ then do mitosis. Mitosis involves abnormal union of cells so what was done in the redistribution step is performed.

2. is in the range $[p_m, p_a)$ then reduction will happen to the cancer cell since these will die due to apoptosis.

3. is in the range $[p_a, p_n)$ then reduction will happen to the cancer cell since these will die due to necrosis and the cancer cell will become necrotic.

Else, the cells are quiescent so nothing happens. This clustering is dependent on the number of necrotic and cancer cells present in the site so each state still has the same probability to happen.

Software Requirements

The necessary programs that were used in order to use CUDA in the study were installed. These programs include:

1. Visual C++ 2008

2. devdriver_3.1_winxp_32_257.21_general

3. cudatoolkit_3.1_win_32

4. gpucomputingsdk_3.1_win_32

The methods were first implemented using OpenGL in C++ for the pure CPU-based implementation. The functions used were eventually converted to CUDA with OpenGL and C++ still used. The CPU used is Intel(R) Core(TM)2 Duo CPU E7500 @ 2.93 GHz and the video card used is NVidia GeForce 9500 GT.

CUDA was used because GPU is specialized for parallel computations, this feature of GPU is what graphics rendering is about [4]. GPU has more transistors for data processing rather than data caching and flow control when compared to CPU. This can be seen in the figures below.

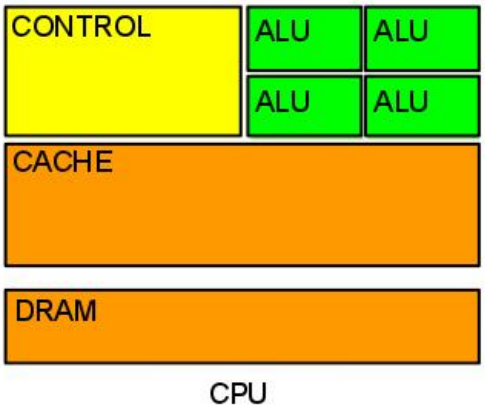

Figure 3: CPU transistor allocation[4].

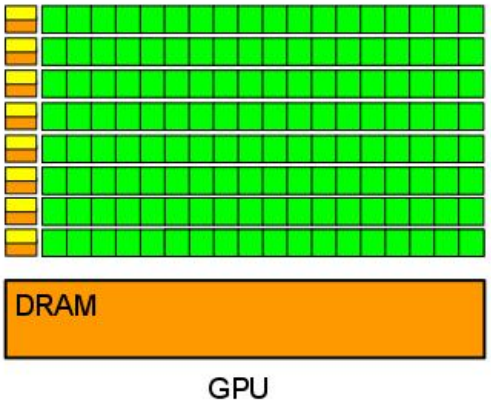

Figure 4: GPU transistor allocation[4].

GPU is good to use in problems that can be expressed in parallel since all the threads involved in CUDA run the same program with high arithmetic intensity. Because the same program is executed for each data element, there is a lower requirement for sophisticated flow control [4].

A relevant data structure used in the implementation is the array. The matrices used in the model were represented as 2D arrays. These arrays were allocated in the GPU through the $cudaMalloc()$ function and copied the contents from the host to the device through the $cudaMemcpy()$ function. In CUDA, two 2D arrays are needed for each matrix, one for

the host and one for the device. The transistors contain grids that contain blocks which in turn contain threads. Since the same program or code is executed by the threads per block, parallelism takes place. Firgure 5 shows this scenario.

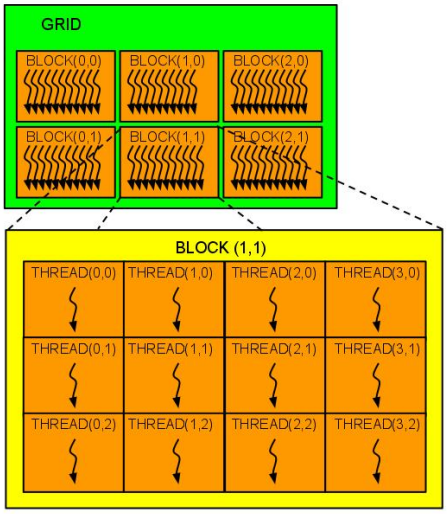

Figure 5: Grid and Block Visualization [4].

In the CPM combined with LGCA CUDA implementation, functions that were translated to CUDA include the *probability* function which computes for the probability of mitosis, apoptosis, necrosis, and quiescent based on energy functions. *Matrix initialization* and *matrix copying* functions were also translated including functions for *canal generation* and the *propagation* of the cells.

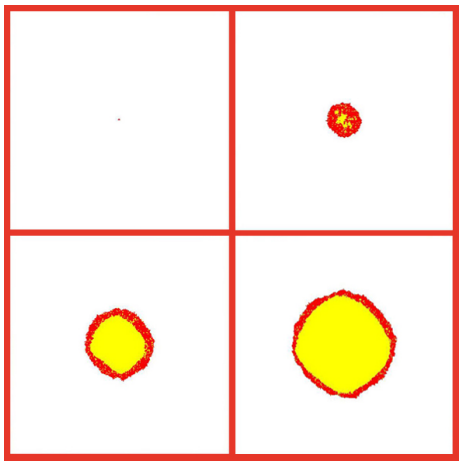

Figure 6: Simulation snapshot at t=0, t=50, t=100, and t=150.

4. RESULTS AND DISCUSSION

A visualization of cancer growth is shown in Figure 6. The simulation was done on a 300x300 matrix where cancer cells were planted in the middle of the matrix. The defined values for the K_{CC}, K_{NC}, and K_{NN} are 3, 1.5, and 3, respectively. These values were taken from the study done by M. Ghaemi and A. Shahrokhi[1] for validation.

Converting a function call into CUDA requires two variables for a particular parameter in a function. The first variable is for the host and the second one for the device. The value used in the function, which is present in the host variable, is copied to the device variable for use as the value in the CUDA function.

The number of blocks defined is simply the N dimension of the array or matrix used; a matrix with a size N requires NxN blocks. The number of threads used was also defined. The number of blocks point towards the function call and the number of threads point towards the device parameters of the function. Then the value of the needed device variable is copied to its partner host variable so that the variables are updated. Finally, all device variables used are freed. Figure 7 shows an example of the CUDA function call used in the implementation.

```
//=============================================================================
//                                PROPAGATION STEP
//=============================================================================

//Allocate device variable to the device
cudaMalloc((void **)&dev_cancer_temp,  MATRIX_DIM_2 * MATRIX_DIM_2 * sizeof(int));
cudaMalloc((void **)&dev_cancer,  MATRIX_DIM_2 * MATRIX_DIM_2 * sizeof(int));
cudaMalloc((void **)&dev_necros_temp,  MATRIX_DIM_2 * MATRIX_DIM_2 * sizeof(int));
cudaMalloc((void **)&dev_necros,  MATRIX_DIM_2 * MATRIX_DIM_2 * sizeof(int));

//copy necessary or needed values from host variables to device variables
cudaMemcpy(dev_cancer,  cancer, MATRIX_DIM_2 * MATRIX_DIM_2 * sizeof(int),
                                cudaMemcpyHostToDevice);
cudaMemcpy(dev_necros,  necros, MATRIX_DIM_2 * MATRIX_DIM_2 * sizeof(int),
                                cudaMemcpyHostToDevice);

//declare the size of the threads and the grid
dim3 thread(MATRIX_DIM_2/13.2,MATRIX_DIM_2/13.2);
dim3 grid(MATRIX_DIM_2/thread.x,  MATRIX_DIM_2/thread.y);
//call the CUDA function
propagate<<<grid, thread>>>(dev_cancer_temp,dev_cancer,dev_necros_temp,dev_necros);

//copy results from the CUDA function to the host variables
cudaMemcpy(cancer_temp,  dev_cancer_temp,  MATRIX_DIM_2 * MATRIX_DIM_2 * sizeof(int),
                                cudaMemcpyDeviceToHost);
cudaMemcpy(necros_temp,  dev_necros_temp,  MATRIX_DIM_2 * MATRIX_DIM_2 * sizeof(int),
                                cudaMemcpyDeviceToHost);

//Cleanup, free the used space in the device
cudaFree(dev_cancer_temp);
cudaFree(dev_cancer);
cudaFree(dev_necros_temp);
cudaFree(dev_necros);
//=============================================================================
//=============================================================================
```

Figure 7: Host code which calls the CUDA function.

In the converted CUDA function, the defined variables x and y are set to be $blockIdx.x$ and $blockIdx.y$ times the $blockDim$ of each and add each $threadIdx$ declared for each variable. These traverse the device of the GPU and use the GPU to process the values that the function performs. The i variable has the value $((MATRIX\_DIM + 2) * x) + y)$ which is the 1D array position value equivalent to a 2D array position (x, y). Figure 8 shows an example of a CUDA function in the implemented program.

Figure 9 shows the growth in number of cancer cells for each time step. Since agents that may stop cancer growth were

```
__global__ void propagate(int *canc_temp, int *canc, int *nec_temp, int *nec){
    //Set the position of the x and y axes in the device using the blocks and threads
    //initialized in the host
    int x = blockIdx.x  * blockDim.x + threadIdx.x;
    int y = blockIdx.y  * blockDim.y + threadIdx.y;

    //Here is the position of the indices of x and y in a 2d array.
    //2d array transformed to a 1d array
    int i = (MATRIX_DIM_2*x)  + y;

    //Do this while x is still not equal to the dimension
    if(x< MATRIX_DIM_2){
            //perform the propagation
            canc_temp[i] = ((canc[i] & 16) |
                            (canc[(MATRIX_DIM_2*(x-1)) + (y)] & 1) |
                            (canc[(MATRIX_DIM_2*(x+1)) + (y)] & 4) |
                            (canc[(MATRIX_DIM_2* (x)) + (y+1)] & 2) |
                            (canc[(MATRIX_DIM_2* (x)) + (y-1)] & 8));
            nec_temp[i] = ((nec[i] & 16) |
                            (nec[(MATRIX_DIM_2*(x-1)) + (y)] & 1) |
                            (nec[(MATRIX_DIM_2*(x+1)) + (y)] & 4) |
                            (nec[(MATRIX_DIM_2* (x)) + (y+1)] & 2) |
                            (nec[(MATRIX_DIM_2* (x)) + (y-1)] & 8));
    }
}
```

Figure 8: *Propagate* function implemented in CUDA.

not included in the model, the graph of the result shows that the cancer cells spread rapidly in time.

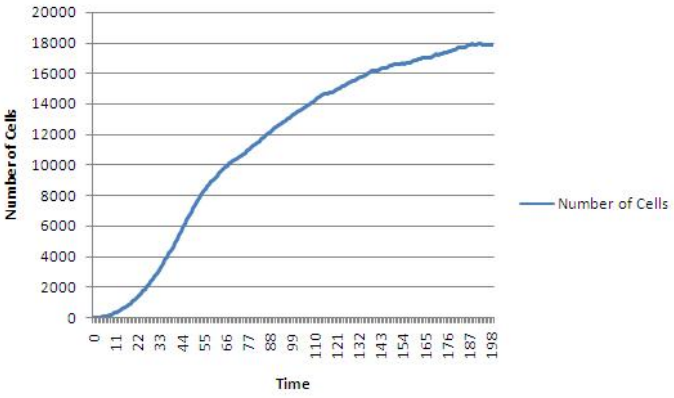

Figure 9: Cancer cell growth.

The average execution time of the program per time step was measured. The program was executed three times and the average of these executions were plotted. The execution time is how long a particular time step is executed by the program in terms of milliseconds (ms) while the time step is the progression or multiplication of cancer cells from only a few to many. In Figure 10, it is shown that the performance of the CPU-based implementation is better than that of CUDA-based for the matrix of size $300x300$.

Other runs were performed for different matrix sizes to check whether it affects the performance. The sizes were $400x400$, $500x500$, and $600x600$. Still, it can be seen in Figures 11-13 that the performance of CPU-based implementation is better than that of the CUDA-based.

5. CONCLUSION

A CUDA-based implementation of the CPM combined with LGCA to model cancer cell growth has been presented. This

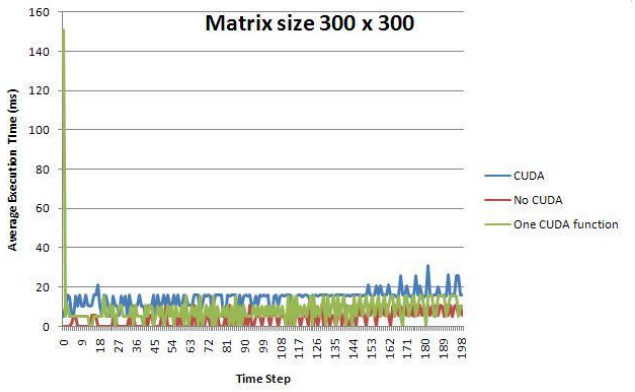

Figure 10: Comparison of the average execution times of the CUDA, CPU, and when only one function of the program (propagation function) was transformed into CUDA with dimension 300x300.

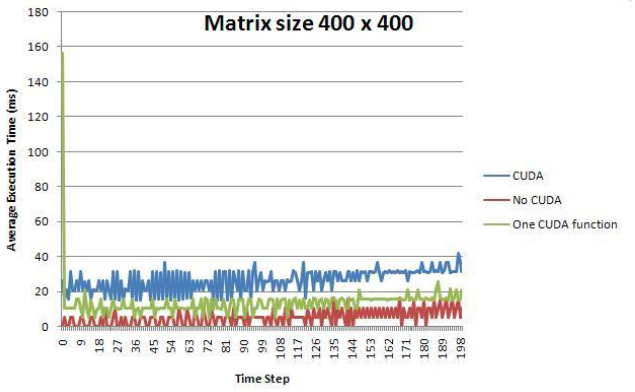

Figure 11: Comparison of the average execution times of the CUDA, CPU, and when only one function of the program (propagation function) was transformed into CUDA with dimension 400x400.

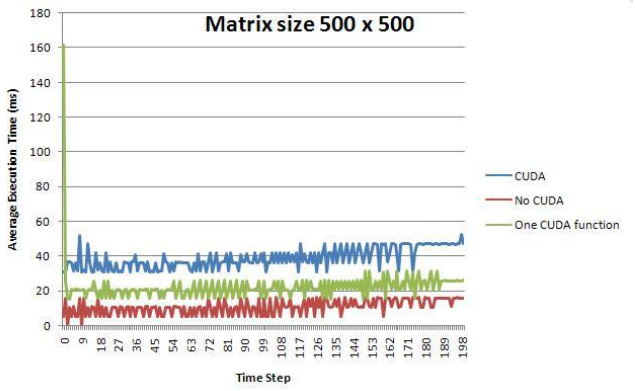

Figure 12: Comparison of the average execution times of the CUDA, CPU, and when only one function of the program (propagation function) was transformed into CUDA with dimension 500x500.

implementation takes advantage of the parallel processing

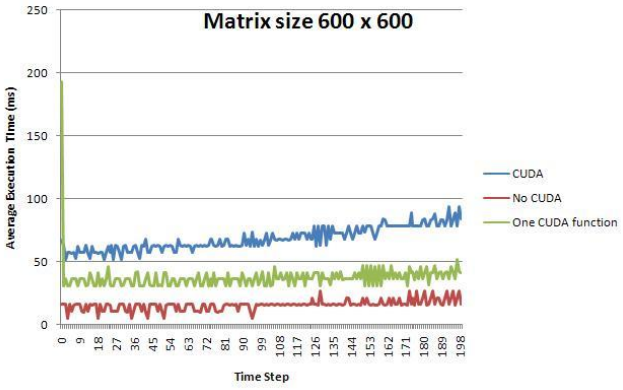

Figure 13: Comparison of the average execution times of the CUDA, CPU, and when only one function of the program (propagation function) was transformed into CUDA with dimension 600x600.

capabilities of the GPU to model cancer cell growth, which is naturally parallel. Based on the results of the simulation, it can be concluded that the CPU-based implementation outperforms the CUDA-based implementation based on the average execution times. Although ideally the CUDA-based implementation should perform better, other factors not covered in this study may have caused the poor performance. One possibility is that the video card used in this study is an old model which might have an effect resulting to the poor performance of the CUDA-based implementation when compared to the CPU-based implementation.

6. RELATED WORK

Several medical mathematical models have already been developed and used by mathematicians due to the need of mathematics to be involved in cancer biology.

Such models included an optimal-control model that provided physicians the best timetables for drug treatments which was studied by Renee Fister [2], a mathematician at Murray State University in Kentucky. Carl Panetta, also a mathematician at Murray State University in Kentucky, used systems of elementary differential equations and predicted the drug response of a patient [2].

A model on brain tumors which used a complex three dimensional brain anatomy and predicted the spread of the tumors was developed by applied mathematicians James Murray and Kristin Swanson from University of Washington [2]. In Israel, A member of the Institute for Medical Biomathematics, Zvia Agur, worked on a model called virtual cancer patient for non-Hodgkins lymphoma. The said model exhibited all the stages of a cancer cells lifecycle [2].

A mathematical model for Chronic Myelogenous Leukemia (CML) and T cell interaction was proposed by Helen Moore and Natasha Li [3] which modeled the interactions and rates of changes between three classifications of cells which were naive T cells, effector T cells, and CML cancer cells using differential equations.

Benjamin Ribba, Thierry Colin and Santiago Schnell also discussed a multiscale mathematical model which investigated "the role of gene-dependant cell cycle regulation in the response of tumors to irradiation therapeutic protocols" [7].

Katarzyna Rejniak and Alexander Anderson introduced Hybrid models of tumor growth [6] which integrated continuous and discrete variables that represented entities like cells involved in cancer cell growth. The integration formed connections between the entities and exhibited cell interactions within tissues. On the other hand, the integration also exhibited the natural interactions between the variables when tumor growth was put under consideration.

Ignacio, et al introduced a Hybrid Discrete Continuum Two-Scale Model which defined that each cell is part of an extracellular matrix and was an individual entity within the said matrix. Interactions within entities were represented by potential functions while the spatial-temporal dynamics were represented using differential functions. These equations characterized the invasion of cancer cells within the matrix [5].

7. ACKNOWLEDGMENTS

The authors would like thank the Institute of Computer Science for the equipment used for this initial study, Dr. Elvira de lara-Tuprio for her help in analyzing the first model used in this study, and the reviewers of this paper.

8. REFERENCES

[1] M. Ghaemi and A. Shahrokhi. Combination of the cellular potts model and lattice gas cellular automata for simulating the avascular cancer growth. *S. El Yacoubi, B. Chopard, and S. Bandini (Eds.): ACRI 2006, LNCS 4173, pp. 297 âĂŞ 303*, 2006.

[2] D. Mackenzie. Mathematical modeling and cancer. *SIAM News*, Jan. 2004.

[3] H. Moore and N. K. Li. A mathematical model for chronic myelogenous leukemia (cml) and t cell interaction. *Journal of Theoretical Biology*, Nov. 2003.

[4] NVIDIA. Nvidia c programming guide. 2010.

[5] I. Ramis-Conde, M. A. Chaplain, and A. R. Anderson. Mathematical modelling of cancer cell invasion of tissuer.

[6] K. A. Rejniak and A. R. A. Anderson. Hybrid models of tumor growth. 2010.

[7] B. Ribba, T. Colin, and S. Schnell. A multiscale mathematical model of cancer growth and radiotherapy efficacy: The role of cell cycle regulation in response to irradiation.