Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 20 (2013) 391 – 398

Procedia Computer Science

Complex Adaptive Systems, Publication 3
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2013- Baltimore, MD

Parallelization of a Bio-Inspired Computational Model for the Simulation of 3-D Multicellular Tissue Growth

Belgacem Ben Youssef*

Department of Computer Engineering, CCIS, King Saud University, Riyadh 11543, Saudi Arabia

Abstract

The use of parallelism may overcome some of the constraints imposed by single processor computing systems. Besides offering faster solutions, applications that are parallelized can solve bigger or more complex problems. For instance, simulations can be run at finer resolutions while physical phenomena can be potentially modeled more realistically. We describe in this paper the development of a bio-inspired parallel algorithm used in the three-dimensional simulation of multicellular tissue growth. We report on the different components of the model where cellular automata is used to model different types of cell populations that execute persistent random walks on the computational grid, collide, and proliferate until they reach confluence. We also discuss the main issues encountered in the parallelization of the model and its implementation on a parallel machine.

Selection and peer-review under responsibility of Missouri University of Science and Technology

Keywords: Bio-inspired simulation model; parallel algorithm; tissue growth; cellular automata

1. Introduction

The demand for computers that are multiple times faster than today's machines continues to grow year after year. The need for such computational power arises, for instance, in the design of better drugs, in the modeling of ecological and biological systems, and in many multidisciplinary applications. One of the ways for increasing the computational power of computers is to use faster and faster components. Until recently, improvements in this area have been extraordinary. In fact, for nearly thirty years, Moore's law has predicted a doubling of transistor capacity every 18-24 months resulting in an average annual increase in processor performance between 25% and 52% [1]. Due to the long memory latency, the decrease in available instruction-level parallelism, and the limitations imposed on power consumption, the increase in processor performance has slowed. This has brought about a "sea change" characterized by a switch from uniprocessors to multiprocessors [2]. Hence, the exploitation of parallelism is

* Corresponding author. Tel.: +966-011-467-3178; fax: +966-011-467-6990.
E-mail address: bbenyoussef@ksu.edu.sa.

Selection and peer-review under responsibility of Missouri University of Science and Technology
doi:10.1016/j.procs.2013.09.292

expected to have more and more significance in the future and users will have to "think in parallel" more than ever before.

Cellular automata (CA) were originally introduced by John von Neumann and Stan Ulam as a possible idealization of biological systems with a particular purpose of modeling biological self-reproduction [3]. CA are dynamic systems, in which both space and time are discrete, consisting of a number of identical cells in a regular lattice (also referred to herein as cellular space/array/grid). Each cell can be in a finite number of states. The next state of each cell is determined, at discrete time intervals, according to its current state, the current state of its neighboring cells, and a next state transition rule or function. Starting from an initial configuration, the cellular array follows a trajectory of configurations defined by the simultaneous application of the local transition rules to all cells in the cellular space. Cellular automata provide a computationally proficient technique for analyzing the collective properties of a network of interconnected cells. Models based on cellular automata provide an alternative approach involving discrete coordinates and variables to represent the complex dynamic system. CA algorithms are also suitable for parallel processing [4-5].

The development of computational models for studying biocomplexity at the cell population and tissue level can provide powerful frameworks in this area [6]. In particular, systems-based approaches can be used to study biocomplexity at the cell population and tissue levels. These approaches consider cells as system components that migrate, proliferate and interact to generate the complex behavior observed in living systems [7-8]. However, employing systems-based approaches could lead to models with high complexity whose solution poses significant computational challenges [9-11]. Computer simulations can be used to shorten the development stage by allowing researchers to quickly screen many alternatives and choose only the most promising ones for laboratory experimentation.

Our previous work in [12] showed that the simulations of realizable multicellular tissue objects is a computationally demanding task that requires small time steps to accurately describe the dynamics of multiple cell populations and long times to complete them. In addition to the size of the cellular array, several input parameters affect the execution time needed to run a simulation, including the initial seeding density, cell migration speed, and cell division time. For instance, these factors combine to yield a serial execution time of over 200 hours for the simulation of a tissue of 1 cm^3 in size. This outcome points to the need for using parallel computing systems to reduce the time to obtain simulation results. The present paper builds on this work by considering the parallelization of a three-dimensional bio-inspired computational and stochastic model for multicellular tissue growth using cellular automata, that accounts for mammalian cell migration, division, and collision. In the next section, we present some related work in this area. Afterwards, we describe the computational model and present the sequential algorithm. We next discuss different aspects related to its parallelization and present the obtained parallel algorithm.

2. Related Work

Various modeling approaches have been used to simulate the population dynamics of proliferating cells. These models can be classified as: *deterministic*, *stochastic*, and based on *cellular automata* or *agents*. Deterministic models, such as the ones developed by Frame and Hu in [13] and Cherry and Papoutsakis in [14], provide insight into simple cell population dynamics. Such models may be useful in fitting specific quantitative results; but they give little or no topological information of the cell colonies before confluence or provide means of interpreting the parameters in terms of the biological processes involved.

Lim and Davies developed a stochastic two-dimensional model based on a matrix of irregular polygons and using the Voronoi tessellation technique to address the issue of cell topology [15]. While this model accounted for the formation and merging of cell colonies, it made some restrictive assumptions on cell interactions and did not address cell motility. Ruaan, Tsai, and Tsao proposed another stochastic model for the simulation of density-dependent growth of anchorage-dependent cells on flat surfaces [16]. Their model included the effects of cell motion and considered that cell sizes varied with time.

A two-dimensional model based on cellular automata was developed by Zygourakis, Bizios, and Markenscoff in [17]. The model allows for contact inhibition during the proliferation process. Using the cellular automata concept, Hawboldt, Kalogerakis, and Behie also modeled cell growth on microcarriers by defining a neighbor table for each cell [18]. Later, Lee et al. showed the importance of cell motility and cell-cell interactions in describing the cell

proliferation rates [19]. This work was succeeded by another model that described the locomotion of migrating endothelial cells in two dimensions [20].

Chang and his team developed a 3-D cellular automata based model to describe the growth of microbial unit cells [21]. This model considered the effects of bacterial cell division and cell death. Other models based on cellular automata have also been used to solve more specific modeling problems. For instance, Kansal et al. developed a model to simulate brain tumor growth dynamics [22]. Their model utilizes a few automaton cells to represent thousands of real cells, thus reducing the computational time requirements of the model while limiting its ability to track individual cells in the cellular space. Another cellular automata model was used by Cickovski et al. in [23] as a framework to simulate morphogenesis. This model used a hybrid approach to simulate the growth of an avian limb. The cellular automaton governed cell interactions while reaction-diffusion equation solvers were used to determine the concentration levels of surrounding chemicals.

There also exist a number of agent-based lattice-free models to simulate tissue growth [24-26]. These models apply the dynamics of cell proliferation and death to describe tissue pattern formation and growth. Related models are suitable for describing the locomotion of a fixed number of cells where cells move relatively slowly with respect to other processes like the diffusion of soluble substances [27-28]. Additional models employ feedback mechanisms between cells and the substrate to model cells entering and leaving the tissue and to establish homeostasis in such systems [29]. Some of the agent-based models use regular triangulation to generate the neighborhood topology for the cells, thus allowing for a continuous representation of cell sizes and locations in contrast to grid-based models [30]. Others utilize multiscale approaches to model collective phenomena in multicellular assemblies [31]. A recent work by Hwang et al. in [9] reviews a number of rule-based modeling techniques of multicellular biological systems with a particular focus on cellular automata and agent-based models.

3. Computational Model

Tissue regeneration is a highly dynamic process. When cells are seeded in a 3-D scaffold, they migrate in all directions, interact with each other and proliferate until they completely fill the space available to them. This assumes that enough nutrients are always available to sustain cell growth everywhere in the interior of the scaffold. To model this dynamic process, we consider cellular automata consisting of three-dimensional grids with a total of N^3 cubic computational sites [3, 32]. Each site is a *finite automaton* that can exist in one of a finite number of states at each time interval. That is, a site may be either:

- empty and available for a cell to move in, or
- occupied by a cell, which is at a given point in its mitotic cycle and moves in a certain direction. No other cell can move or divide into an already occupied site.

Proliferating cells execute *persistent random walks* in space [20, 33]. This process consists of the following stages:

1. Each cell in the population moves in one direction for a certain period of time (*persistence*). At the end of this interval, the cell stops and turns to continue its migration in another direction. The persistence is a random variable whose density function can be determined experimentally.
2. When two cells *collide*, they stop for a short period of time before resuming their migration to move away from each other.
3. At the end of its cycle, a cell stops to divide into two daughter cells. The cell cycle or *division time* is another random variable whose density function can be experimentally measured.
4. This process is repeated until the cell population has completely filled the scaffold or until the cells cannot migrate and divide any further.

To simulate these dynamics, the state $x_i(t)$ of each cellular automaton takes values from a set of integer numbers that code all the required information about the cell type, its migration speed, the direction of movement, and the time remaining until the next direction change and the next cell division. Our model also considers that cell division time is a random variable whose probability density function can be obtained experimentally using the procedure described by Lee and coworkers [34]. Hence, every automaton has its state evolving at discrete time steps Δt

through interactions with neighboring automata. Let us consider the *j*-th automaton that contains a cell, of a given type, at time t^r. Its state $x_j(r)$ is specified by the following numbers:

1. Cell type index $k_{t,j}$: For each cell population, this is a unique identifier. The number of modeled cell populations is based on the number of digits used to represent $k_{t,j}$. Using a single digit, then up to nine different cell populations can be simulated with each having its own division and migratory parameters.
2. Migration index m_j: If $m_j = 1, 2, \ldots, 6$, then the cell is migrating in one of the six directions (east, north, west, south, up and down). If $m_j = 0$, the cell is stationary.
3. Division counter $k_{d,j}$: The time that must elapse before the cell divides is equal to $t_d = k_{d,j}\Delta t$. For each iteration, this counter is decremented by one and the cell divides when $k_{d,j} = 0$.
4. Persistence counter $k_{p,j}$: The time that must elapse before the cell changes its direction of movement is equal to $t_p = k_{p,j}\Delta t$. For each iteration, this counter is decremented by one and the cell turns when $k_{p,j} = 0$.

4. Sequential Algorithm

4.1. Initial condition

The sites that will be occupied by cells at time t^0 are selected. The assignment of seed cells to the grid sites may be done either randomly (using, for example, a uniform distribution) or according to rules that emulate special cases of tissue regeneration as in wound healing [35]. Afterwards, an initial state $x_j(0)$, at time t^0, is assigned to each occupied site *j* based on the population characteristics of that cell type. The migration index m_j is randomly selected, the value of the persistence counter $k_{p,j}$ is properly chosen, and the cell division counter $k_{d,j}$ is set according to the experimentally determined distribution of cell division times. The integer counters $k_{p,j}$ and $k_{d,j}$ will be decremented at every iteration and the cell will change its direction of movement or divide when $k_{p,j} = 0$ or $k_{d,j} = 0$, respectively.

4.2. Iterative operations

At each time instant $t^r = t^{r-1} + \Delta t, \quad r = 1, 2, \ldots$

- **Step 1**: Randomly select a computational site.
- **Step 2**: If this site contains a cell that is ready to divide, execute the *division routine* and go to step 5.
- **Step 3**: If this site contains a cell that is ready to change its direction of movement, execute the *direction change routine* and go to step 5.
- **Step 4**: Otherwise, try to move the cell to a neighboring site in the direction indicated by the migration index of its current state:
    - If this site is free, *mark* it as the site that will contain the cell at the next time step and decrement the persistence and cell phase counters by one.
    - If this site is occupied, we have a cell-cell collision. The cell will remain at the present site (thus, entering the *stationary state*) and execute the *direction change* routine after a pre-specified number of iterations.
- **Step 5**: Select another site and repeat steps 2–4 until all sites have been examined.
- **Step 6**: Update the states of all sites so that the locations of all cells are set for the next time step.

Due to space limitations, the division and direction change routines are described elsewhere [12].

5. Parallelization

The parallel algorithm we designed to simulate the dynamics of multicellular tissue growth was implemented on a distributed-memory cluster machine using the Message Passing Interface (MPI) [36-37]. Our discussion here will focus on the main issues we faced during the different steps comprising the parallelization task. We begin by looking at the architecture of the application at hand. Our application falls in the category of *loosely synchronous* applications as categorized by Pancake in [38]. Such applications exhibit certain characteristics where the amount of computation could vary from one partition and time step to another because it depends on the amount of useful data,

which is proportional to the number of occupied sites [38]. A single processor experiences different workloads from the early time steps to the later ones as cells divide. The need to exchange data among processors necessitates that each processor be able to determine when the other nodes are ready for this exchange so that data not yet used are not overwritten. Between these exchange points, the different nodes proceed at their own rates. Since the workload now varies both temporally and spatially, much effort must be spent to evenly distribute it among the nodes. In order to minimize overhead, we used a *static* and cell distribution-dependent *load-balancing* strategy whereby each node stays responsible for a fixed part of the cellular space. This is known as the *Eulerian* method [39]. Static methods are uncomplicated, but can have difficulty handling subsequent load imbalances. The major advantage of using static load balancing is that the entire overhead of the load-balancing process is incurred at compile time. This represents a one-time and fixed cost that results in more efficiency.

Further, we were interested in solutions which are easily scalable, that is, the solutions should be efficient for a wide range of number of processors, with the goal of minimizing the overall execution time of the program, while minimizing the communication delays. Our choice of static strategies was based on the fact that the behavior of cells (their division and migration) is random. We observed that the computational load fluctuations between neighboring sub-domains tend to average out, thus maintaining a load-balanced computation. This means that the number of cells leaving a sub-domain is counterbalanced by a nearly equal number of cells entering it. Moreover, the choice of a cell distribution-dependent load-balancing technique fulfilled the twin objective of not only conserving the required load balancing but also that of maintaining the efficiency of the parallel computation. Below, we list the main steps of the parallelization:

- **Domain Decomposition**: We used a seeding mode where cells are uniformly and randomly distributed in the cellular array. This is widely known as the most common seeding mode in tissue engineering applications [40]. This type of distribution is amenable to a slab decomposition that can be achieved by dividing the cellular array into slabs along the z dimension. Here, the area of the boundary between any two sub-domains is equal to $N_x \times N_y$ sites. Hence, the maximum amount of data communicated from a sub-domain to its neighbors is $2 \times N_x \times N_y$ data elements at any given instant.
- **Mapping and Tuning**: To implement the slab decomposition, the sub-domains were logically mapped onto a one-dimensional processor grid representing a topology of a linear array. In our application, this topology reflects the logical communication pattern of the processes.
- **Dealing with Correctness**: Our objective is to solve the same problem with the parallel code as with the serial code [35]. *Correctness* for the scope of our application is related to the kind of problem our code solves, not to the exact results we get from the machine. This means that two algorithms A1 and A2 may give different cell population distributions after a certain number of steps, but both distributions could be correct solutions to the problem if they were achieved according to the same CA rules described earlier.
- **Parallelization of the Random Number Generator**: Our serial algorithm uses often a pseudo-random number generator (PRNG) to select a site on the grid and to reset the state of a site occupied by a living cell. A parallel implementation of the PRNG on one node would be very inefficient and would constitute a bottleneck. Instead, we parallelized the random number generator in a way that keeps our computation deterministic. We employed the leap-frog strategy which interleaves the parallel subsequences of random numbers, reducing them to a single sequence that is nearly equivalent to the sequential one [41].
- **Splitting Cell Movement and Division**: Performing cell movement and division in one step will lead to correctness problems. The solution is to use a "splitting" technique that consists of two aggregate steps: the first computes the next position of a cell or its daughter cell, while the second step is the one that actually moves/divides the cell and updates the state [42]. The sequential implementation inherently uses this splitting technique so that there is no possibility of conflicts. To preserve the semantics of the serial algorithm, the parallel algorithm must communicate with the neighboring processors between the first and second step. If we do not communicate then, we lose correctness since this leads to having more than one cell occupy a given site, which is a violation of one of the rules of our cellular automaton.

6. Parallel Algorithm

The goal of this parallel algorithm is to reduce the amount of communication among processes by exchanging shared boundaries during a simulation time step *only* when a process has calculated the movement/division of all

cells in the sub-domain and not each time a cell attempts to cross over to a neighboring sub-domain. Thus, for cells attempting to cross over to a neighboring sub-domain, their inquiries are recorded and sent to the corresponding neighbor, after all the cells of the sub-domain have been considered. The actual movement/division of a cell that crosses over to a neighboring sub-domain is performed after the exchange of the shared boundaries. Using the current processor as a reference point, we define the following terms and notations that will be used in describing the steps of the parallel algorithm:

- **Myself**: Identifies the id number of the current processor (for example, id i).
- **Mypred**: Identifies the id number of the predecessor processor using a logical linear array topology (for example, id $i-1$).
- **Mysucc**: Identifies the id number of the successor processor using a logical linear array topology (for example, id $i+1$).
- **Volume Coverage**: Represents the percentage of occupied sites. A value of 99.99% is usually chosen.
- **X(N_x, N_y, N_z)**: The global cellular array containing all cells.
- **X(N_x, N_y, 0:n_z+1)**: Part of the cellular array owned by the current processor (local sub-domain) including two ghost layers to accommodate shared boundaries with the two neighboring processors, where $n_z = N_z/P$.
- $\mathbf{M}_{\text{crossing\_to\_mypred}}$: A 2-D matrix containing the state information of all cells attempting to cross over from the bottom layer (layer 1) of current processor to the top layer (layer n_z) of predecessor processor.
- $\mathbf{M}_{\text{crossing\_to\_mysucc}}$: A 2-D matrix containing the state information of all cells attempting to cross over from the top layer (layer n_z) of current processor to the bottom layer (layer 1) of successor processor.
- $\mathbf{M}_{\text{crossing\_from\_mypred}}$: A 2-D matrix containing the state information of all cells attempting to cross over from the top layer (layer n_z) of predecessor processor to the bottom layer (layer 1) of current processor.
- $\mathbf{M}_{\text{crossing\_from\_mysucc}}$: A 2-D matrix containing the state information of all cells attempting to cross over from the bottom layer (layer 1) of successor processor to the top layer (layer n_z) of current processor.
- $\mathbf{M}_{\text{rejected\_to\_mypred}}$: A 2-D matrix containing the position and state information of all cells rejected by current processor and going back to predecessor processor.
- $\mathbf{M}_{\text{rejected\_to\_mysucc}}$: A 2-D matrix containing the position and state information of all cells rejected by current processor and going back to successor processor.
- $\mathbf{M}_{\text{rejected\_by\_mypred}}$: A 2-D matrix containing the position and state information of all cells rejected by the predecessor processor and going back to current processor.
- $\mathbf{M}_{\text{rejected\_by\_mysucc}}$: A 2-D matrix containing the position and state information of all cells rejected by the successor processor and going back to current processor.
- **Layer** 0: Represents layer n_z of the predecessor processor.
- **Layer** n_z+1: Represents layer 1 of the successor processor.

We present in Figure 1 the actions of an even-numbered process P_{2i} of the parallel algorithm during the k^{th} simulation time step. Unless otherwise specified, “send” and “receive” in the pseudo-code mean “MPI_ISEND” and “MPI_IRECV”, respectively [37].

7. Conclusion and Future Work

We have presented in this paper the parallelization of our three-dimensional simulation model for multicellular tissue growth including a description of the main issues dealt with during this task. As part of our future work, we will focus on extending this model to include cell differentiation and cell death as well as its implementation on other parallel systems such as multi-core machines. We will also work on integrating a visualization solution with the simulation model to assist researchers to explore the spatial and temporal domains of tissue growth in real time and to provide them with useful and insightful means to interpret and analyze simulation data.

Acknowledgements

The author would like to gratefully acknowledge the support for this research work provided by the Research Center in the College of Computer & Information Sciences (under Project Number: RC121231) and the Deanship of Scientific Research, both at King Saud University, Riyadh, Saudi Arabia.

Inputs: Myself, mypred, mysucc, volume coverage, $X(N_x, N_y, 0{:}n_z{+}1)$ at the end of $(k\text{-}1)^{st}$ time step.
Output: $X(N_x, N_y, 0{:}n_z{+}1)$ at the end of k^{th} time step.

```
While the specified volume coverage has not been reached
Do
    While not all cells in the sub-domain have been considered
    Do
        Select the next cell in the sub-domain according to a random order;
        Calculate cell movement/division;
        If the cell is not crossing over to a neighboring sub-domain
        Then
            Execute its movement/division;
        Else
            Record the cell's current position and the calculated new cell state
            1) in a matrix M_crossing_to_mypred for cells attempting to cross over from layer 1 of
               myself to layer n_z of mypred and
            2) in a matrix M_crossing_to_mysucc for cells attempting to cross over from layer n_z of
               myself to layer 1 of mysucc;
        EndIf
        Send M_crossing_to_mypred to mypred and M_crossing_to_mysucc to mysucc;
        Receive M_crossing_from_mypred from mypred and M_crossing_from_mysucc from mysucc;
        For each cell recorded in M_crossing_from_mypred and M_crossing_from_mysucc
        Do
            1) Decide whether a cell that crosses over from a neighbor will be accepted (into an
            empty site of the sub-domain) or rejected due to having more than one cell moving
            to a given site;
            2) Record the position of a rejected cell in a matrix M_rejected_to_mypred for cells going
            back to mypred and in a matrix M_rejected_to_mysucc for cells going back to mysucc;
        EndDo
        Send M_rejected_to_mypred to mypred and M_rejected_to_mysucc to mysucc;
        Receive M_rejected_by_mypred from mypred and M_rejected_by_mysucc from mysucc;
        For each cell recorded in M_rejected_by_mypred and M_rejected_by_mysucc
        Do
            Recalculate cell movement/division from the cell's original position;
            Execute cell movement/division;
        EndDo
    EndDo
    Send layer 1 to mypred and layer n_z to mysucc;
    Receive layer 0 from mypred and layer n_z+1 from mysucc;
    Update the time step of the simulation;
EndDo
```

Fig. 1. The parallel algorithm showing actions performed by an even-numbered process P_{2i} during the k^{th} simulation time step

References

1. Hennessy, J. L. and Patterson, D. A. 2012. *Computer architecture: A quantitative approach*. Morgan Kaufmann Publishers, fifth edition.
2. Brodtkorb, A. R., Dyken, C., Hagen, T. R., Hjelmervik, J. M., and Storaasli, O. O. 2010. State-of-the-art in heterogeneous computing. *Sci. Prog.* 18, 1, 1-33.
3. Wolfram, S. 1994. Cellular automata and complexity: Collected papers. Addison-Wesley.
4. Chaudhuri, P. P., Chowdhury, D. R., Nandi, S., and Chattopadhyay, S. 1997. *Additive cellular automata: Theory and applications*. Volume 1, IEEE Computer Society Press, Los Alamitos, CA.
5. Deutsch, A. and Dormann, S. 2005. *Cellular automaton modeling of biological pattern formation: Characterization, applications, and analysis*. Springer-Verlag, Boston.
6. Lim, J. H. F. and Davies, G. A. 1990. A stochastic model to simulate the growth of anchorage-dependent cells on flat surfaces. *Biotechnol. Bioeng.* 36, 547-62.
7. Majno, G. and Joris, I. 2004. *Cells, tissues and disease: Principles of general pathology*. Oxford University Press, Oxford, UK.
8. Page, E. H. and Nance, R. E. 1994. Parallel discrete event simulation: A modeling methodological perspective. In *Proceedings of the 1994 Workshop on Parallel and Distributed Simulation*, 88-93.

9. Hwang, M., Garbey, M., Berceli, S. A., and Tran-Son-Tay, R. 2009. Rule-based simulation of multi-cellular biological systems-A review of modeling techniques. *Cell. Mol. Bioeng*. 2, 3, 285-94.
10. Lauffenburger, D. A. and Linderman, J. J. 1993. *Receptors: Models for binding trafficking and signaling*. Oxford University Press, NY.
11. Levin, S. A., Grenfell, B., Hastings, A., and Perelson, A. S. 1997. Mathematical and computational challenges in population biology and ecosystems science. *Sci*. 275, 5298, 334-43.
12. Ben Youssef, B. and Tang, L. Simulation of multiple cell population dynamics using a 3-D cellular automata model for tissue growth. *International Journal of Natural Computing Research*, 1, 3, 1-18, 2010.
13. Frame, K. K. and Hu, W. S. 1988. A model for density-dependent growth of anchorage-dependent mammalian cells. *Biotechnol. Bioeng*. 32, 1061-66.
14. Cherry, R. S. and Papoutsakis, E. T. 1989. Modelling of contact-inhibited animal cell growth on flat surfaces and spheres. *Biotechnol. Bioeng*. 33, 300-5.
15. Lysaght, M. J. and Hazlehurst, A. L. 2004. Tissue engineering: The end of the beginning. *Tissue Eng*. 10, 1-2, 309-20.
16. Ruaan, R. C., Tsai, G. J., and Tsao, G. T. 1993. Monitoring and modeling density-dependent growth of anchorage-dependent cells. *Biotechnol. Bioeng*. 41, 380-89.
17. Zygourakis, K., Bizios, R., and Markenscoff, P. 1991. Proliferation of anchorage-dependent contact-inhibited cells: I. Development of theoretical models based on cellular automata. *Biotechnology & Bioeng*ineering, 38, 5, 459–70.
18. Hawboldt, K. A., Kalogerakis, N., and Behie, L. A. 1994. A cellular automaton model for microcarrier cultures. *Biotechnol. Bioeng*. 43, 1, 90-100.
19. Lee, Y., Kouvroukoglou, S., McIntire, L. V., and Zygourakis, K. (1995). A cellular automaton model for the proliferation of migrating contact-inhibited cells. *Biophys. J.* 69, 10, 1284-98.
20. Lee, Y., Markenscoff, P., McIntire, L. V., and Zygourakis, K. 1995. Characterization of endothelial cell locomotion using a Markov chain model. *Biochem. Cell Biol*. 73, 461-72.
21. Chang, L., Gilbert, E. S., Eliashberg, N., and Keasling, J. D. 2003. A three-dimensional, stochastic simulation of biofilm growth and transport-related factors that affect structure. *Microbiol*. 149, 10, 2859-71.
22. Kansal, A. R., Torquato, S., Harsh IV, G. R., Chiocca, E. A., and Deisboeck, T. S. 2000. Simulated brain tumor growth dynamics using a three-dimensional cellular automaton. *J. Theor. Biol*. 203, 4, 367-82.
23. Cickovski, T. M., Huang, C., Chaturvedi, R., Glimm, T., Hentschel, H. G. E., Alber, M. S., Glazier, J. A., Newman, S. A., and Izaguirre, J. A. 2005. A framework for three-dimensional simulation of morphogenesis. *IEEE/ACM T. Comput. Biol. Bioinformat*. 2, 4, 273-88.
24. Drasdo, D., Kree, R., and McCaskill, J. S. 1995. Monte Carlo approach to tissue-cell populations. *Phys. Rev. E*. 52, 6, 6635-57.
25. Palsson, E. 2001. A three-dimensional model of cell movement in multicellular systems. *Future Gener. Comp. Sy*. 17, 835-52.
26. Schaller, G. and Meyer-Hermann, M. 2005. Multicellular tumor spheroid in an off-lattice voronoi-DeLaunay cell model. *Phys. Rev. E*. 71, 5 Pt 1, 051910.
27. Beyer, T. and Meyer-Hermann, M. Delauny object dynamics for tissues involving highly motile cells. In *Cell mechanics: From single scale-based models to multiscale modeling*, A. Chauviere, L. Preziosi, and C. Verdier, Eds. CRC Press, 417-42, 2010.
28. Jiang, Y., Levine, H., and Glazier, J. 1998. Possible cooperation of differential adhesion and chemotaxis in mound formation of Dictyostelium. *Biophys. J.* 75, 6, 2615-25.
29. Fu, Y. X. and Chaplin, D. D. 1999. Development and maturation of secondary lymphoid tissues. *Annu. Rev. Immunol*. 17, 399-433.
30. Beyer, T., Schaller, G., Deutsch, A., and Meyer-Hermann, M. 2005. Parallel dynamic and kinetic regular triangulation in three dimensions. *Comput. Phys. Commun*. 172, 2, 86-108.
31. Drasdo, D., Jagiella, N., Ramis-Conde, I., Vignon-Clemental, I. E., and Weens, W. 2010. Modeling steps from benign tumor to invasive cancer: Examples of intrinsically multiscale problems. In *Cell mechanics: From single scale-based models to multiscale modeling*, A. Chauviere, L. Preziosi, C. Verdier, Eds., CRC Press, 379-416.
32. Tchuente, M. 1987. Computation on automata networks. In *Automata networks in computer science: Theory and applications*, F. Fogelman-Soulie, Y. Robert, M. Tchuente, Eds. Princeton University Press, Princeton, NJ.
33. Cheng, G., Ben Youssef, B., Markenscoff, P., and Zygourakis, K. 2006. Cell population dynamics modulate the rates of tissue growth processes. *Biophys. J*. 90, 3, 713-24.
34. Lee, Y., McIntire, L. V., and Zygourakis, K. 1994. Analysis of endothelial cell locomotion: Differential effects of motility and contact inhibition. *Biotechnol. Bioeng*. 43, 7, 622-34.
35. A.F. Marée, P. Hogeweg, How amoeboids self-organize into a fruiting body: Multicellular coordination in Dictyostelium Discoideum. *P. Natl. Acad. Sci. USA*. 98, 7, 3879-83.
36. Fox, G. C., Williams, R. D., and Messina, P. C. 1994. *Parallel computing works!* Morgan Kaufmann Publishers, Inc.
37. Quinn, M. J. 2004. *Parallel programming in C with MPI and OpenMp*. McGraw-Hill, Dubuque, IA.
38. Pancake, C. M. 1996. Is Parallelism for You? *IEEE Comput. Sci. Eng*. 3, 2, 18-37.
39. van Hanxleden, R. and Scott, L. R. 1991. Load balancing on message passing architectures. *Journal of Parallel and Distributed Computing,* 13, 3, 312 -24.
40. Chung, C. A., Lin, T-H., Chen, S-D., and Huang, H-I. 2010. Hybrid cellular automaton modeling of nutrient modulated cell growth in tissue engineering constructs. *J. Theor. Biol*. 262, 2, 267-78.
41. Fox, G. C., Johnson, M. A., Lyzenga, G. A., Otto, S. W., Salmon, J. K., and Walker, D. W. 1988. *Solving problems on concurrent processors: Volume I. General techniques and regular problems*. Prentice Hall, Englewood Cliffs, NJ.
42. Hoshino, T., Hiromoto, R., Sekiguchi, S., and Majima, S. 1989. Mapping schemes of the particle-in-cell method implemented on the PAX computer. *Parallel Comput*. 9, 1, 53-75.