Scientific Research

A High-Performance Cellular Automaton Model of Tumor Growth with Dynamically Growing Domains

Jan Poleszczuk[1], Heiko Enderling[2]

[1]College of Inter-Faculty Individual Studies in Mathematics and Natural Sciences, University of Warsaw, Warsaw, Poland
[2]Integrated Mathematical Oncology, H. Lee Moffitt Cancer Center and Research Institute, Tampa, USA
Email: j.poleszczuk@mimuw.edu.pl, heiko.enderling@moffitt.org

ABSTRACT

Tumor growth from a single transformed cancer cell up to a clinically apparent mass spans many spatial and temporal orders of magnitude. Implementation of cellular automata simulations of such tumor growth can be straightforward but computing performance often counterbalances simplicity. Computationally convenient simulation times can be achieved by choosing appropriate data structures, memory and cell handling as well as domain setup. We propose a cellular automaton model of tumor growth with a domain that expands dynamically as the tumor population increases. We discuss memory access, data structures and implementation techniques that yield high-performance multi-scale Monte Carlo simulations of tumor growth. We discuss tumor properties that favor the proposed high-performance design and present simulation results of the tumor growth model. We estimate to which parameters the model is the most sensitive, and show that tumor volume depends on a number of parameters in a non-monotonic manner.

KEYWORDS

Cellular Automaton; Dynamic Boundaries; Tumor Model; Cancer Stem Cells; Sensitivity Analysis

1. Introduction

Simulating complex multi-scale cellular automata is still a great challenge despite advances in computational power of modern computers in recent years. Cellular automata are increasingly used to simulate tumor growth dynamics [1-15]. Whilst many efficient ways exist to simulate deterministic and synchronous cellular automata, such as Conway's "Game of Life" [16], high-performance simulation of stochastic cancer cell kinetics and emerging multi-scale tumor population dynamics is still in is infancy. In stochastic Monte Carlo cancer models, cells are not governed by simple deterministic rules but by probability distributions of coupled internal states and non-trivial interactions with the continuously changing local environment. Additionally, tumor population dynamics emerge from the interaction of millions of cells, and often the development of such populations from few initial cells needs to be simulated. This poses problems of bridging many temporal and spatial scales. Due to the stochastic nature of single cell kinetics many simulations for the same scenario need to be performed in order to obtain averaged and statistically significant results. To further complicate matters, in typical tumor growth models many parameters need to be estimated in high-dimensional parameter sweeps and sensitivity analysis needs to be performed to study parameter influence on overall dynamics.

The main advantage of utilizing cellular automata in cancer modeling is the ability to formalize experimentally observable single-cell kinetics [17,18] and observe emerging population level dynamics without a priori knowledge of tumor behavior. Because of their apparent resemblance of *in vitro* cell culture models, cellular automata may be referred to as *in silico* experiments [19]. Automata simulations enable visualization, measurement and perturbation of cell kinetics as well as their interaction with the environment. Herein we describe a

simple cellular automaton tumor growth model, and discuss computer memory access, data structures, domain setup and implementation techniques that enable high performance multi-scale simulations.

2. Tumor Growth Model

A cancer cell is an individual entity that occupies a single grid point of $(10 \; \mu m)^2$ on a two-dimensional square lattice. Each cancer cell is characterized by its specific trait vector [cct, ρ, μ, α] denoting cell cycle time, proliferation potential, migration potential and rate of spontaneous death, respectively [20]. We assume a heterogeneous tumor population consisting of so-called cancer stem cells and non-stem cancer cells. Cancer stem cells are assumed to be immortal and have unlimited proliferation potential (*i.e.*, $\alpha = 0$, $\rho = \infty$), whereas non-stem cancer cells can only divide a limited number of times ρ_{max} before cell death. Each cell type can divide symmetrically to produce two daughter cells with parental phenotype. Both populations are coupled through asymmetric division of cancer stem cells, which with probability $1 - p_s$ (where p_s is the probability of symmetric cancer stem cell division) produces a cancer stem cell and a non-stem cancer cell, which inherits an initial proliferation potential $\rho = \rho_{max}$ that decreases with each subsequent non-stem cell division (**Figure 1(a)**). Cells need adjacent space for migration and proliferation, and cells that are completely surrounded by other cells (eight on a two-dimensional lattice; Moore neighborhood) become quiescent (**Figure 1(b)**). In unsaturated environments, cells proliferate and migrate into vacant adjacent space at random. At each simulation step, cells can undergo spontaneous death with rate α and will be instantaneously removed from the system.

Time is advanced at discrete time intervals $\Delta t = 1/24$ day (*i.e.*, 1 hour), and 24 simulation steps represent one day. At each simulation step, cells are considered in random order to minimize lattice geometry effects and the behavior of each cell is updated. Cell proliferation, migration and death are random events with the respective probabilities scaled to the simulation time step. Cell proliferation and migration are temporally mutually exclusive events. We assume that cells proliferate at each simulation step with probability $p_d = (24/cct) \times \Delta t$, migrate with probability $(1 - p_d)p_m$ and die with probability α. Let $p_m = \mu \times \Delta t$ where μ denotes cancer cell motility speed.

3. Implementation

3.1. Memory Architecture and Data Access

High-performance simulations require fast access to available memory and cached data. How memory is handled depends heavily on simulation design as well as used data structures and procedures. The memory in modern desktop PCs has three layers: the built-in cache memory has the fastest access time (1 - 20 ns) but a very limited size; random access memory (RAM) is slower (50 - 100 ns) but much larger; and hard disk drives (HDD) whilst having large memory have the slowest access time (5 - 10 ms) (**Figure 2**).

State-of-the-art processors may have up to 24 Megabytes (MB) of cache memory and can access up to 4096 Gigabytes (GB) of RAM (c.f., Intel Xeon E7-8830). Cache memory stores the most frequently used RAM locations to reduce access time to necessary information [21]. Due to limited memory size, cached content constantly changes throughout simulations. Simulation time decreases when the *spatial locality* property is unsatisfied, *i.e.* the CPU frequently requires access to information that is not stored in cache memory locations. If a so-called *cache miss* occurs data needs to be retrieved from much slower RAM or even HDD memory. High frequencies
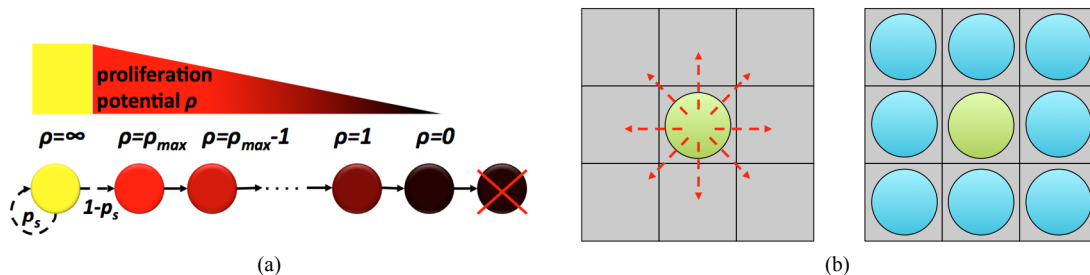

(a)   (b)

Figure 1. (a) Cancer cell lineage; yellow: cancer stem cell, red-black: non-stem cancer cells with decreasing proliferation potential ρ; (b) Tumor cells populate the computational lattice by cell migration and cell proliferation. A cell can randomly migrate to or place a daughter cell into one of the eight adjacent lattice points subject to availability. A cell becomes quiescent if all adjacent lattice points are occupied.
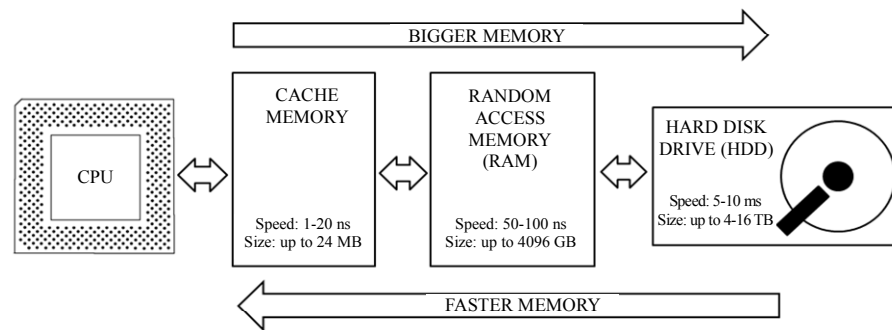
Figure 2. Typical architecture of modern desktop PCs. Central processing unit (CPU) reads memory directly from the fastest cache, which, if the data is unavailable reads from the slower random access memory (RAM) and if needed from the biggest but slowest hard disk drive (HDD).

of *cache misses* dramatically reduce computation speed and optimized algorithms should minimize *cache miss* events.

Let us consider a two-dimensional rectangular lattice coded by a two-dimensional array as commonly used in cellular automata. As computer memory is arranged linearly, higher-dimensional arrays are stored row after row (or column after column). Therefore, if an array element is accessed only parts of its immediate spatial neighborhood will be stored in the cache. Especially for large lattices, 2 of 4 neighbors (2-D von Neumann neighborhood) or 6 of 8 neighbors (2-D Moore neighborhood) are *cache missed*. Whilst convenient at implementation, frequent access of cell neighbors in two- and three-dimensional arrays is memory inefficient and slow.

3.2. Population Geometry and Data Type Optimization

Which data structures are the best to use depends on the cellular processes that are considered as well as the geometry of the emerging population. Prostate tumors, for example, have a very dense, compact structure whereas glioblastoma brain tumors are highly diffusive. Such density difference may be represented by the number of cells on the computational lattice per area or volume. Let us define a dense tumor as a population of cells where each lattice point is occupied by a cell with probability p = 0.99 (*i.e.*, 99%) and a diffusive tumor occupies lattice points with p = 0.5. For cells to migrate or proliferate adjacent lattice points need to be vacant. Dependent on the expected tumor density—either many or few neighbors for most cells—the most efficient data structure for obtaining vacant neighbor lattice sites might be different. To determine cell neighborhood vacancies, a simple array keeping *Boolean* information about lattice points occupied by cells will be highly inefficient for dense tumors. Let us consider morphological erosion, where each cell that is not completely surrounded by other cells is removed from the lattice. For dense population geometries, a coded array containing information about number of vacant spots in the cell neighborhood may be more efficient as unsuccessful scanning of each neighboring lattice point for vacancy is be avoided (**Figure 3(a)** and **(b)**). Appropriate use of C++ data type *char* will not introduce a memory tradeoff as both *char* and *Boolean* require one byte of memory. Using intuitive *int* instead of *char* will require four times more memory and increases computation time as fewer information can be stored in cache memory. A computationally expensive drawback of a coded lattice is the requirement to update all neighboring lattice codes when occupancy of a single grid point changes, which makes this approach less efficient for diffusive tumors (**Figure 3(c)**).

3.3. Random Neighbor Selection

Monte Carlo tumor growth simulations frequently require obtaining a free neighboring lattice site at random, for example for migration or proliferation. A naïve approach may consider all neighboring lattice sites, store those that are vacant in a temporary vector and then select a vector element at random. Alternatively, neighboring lattice sites may be addressed in random order and the first encountered vacant position is selected (**Figure 4(a)**). This simple alternative random access method significantly decreases simulation time in dense (**Figure 4(b)**) and diffusive tumors (**Figure 4(c)**) with increasing lattice size (free spot is selected iteratively for each cell on the lattice). While the naïve procedure is much slower for diffusive tumors as more vacant lattice sites have to be stored in temporary vectors, the alternative random access approach performs equally well irrespective of tumor type.
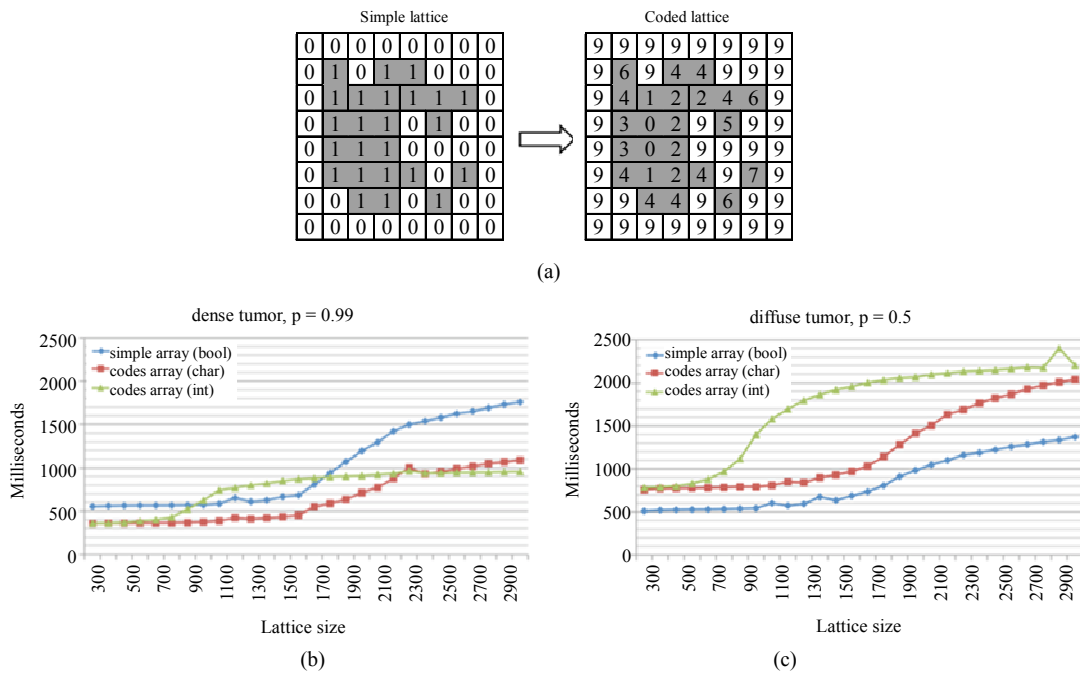
Figure 3. Morphological erosion on a simple Boolean array and a coded lattice. (a) Illustration of transformation from simple Boolean lattice into coded lattice. In a coded lattice each occupied grid point contains the information about the number of free spots in its neighborhood, and 9 represents an empty grid point; (b) and (c) Comparison of the evaluation times for dense tumors (b) and diffuse tumors (c). Three considered data structures are simple Boolean array (blue diamonds) and coded lattices using char (red squares) or integer values (green triangles).
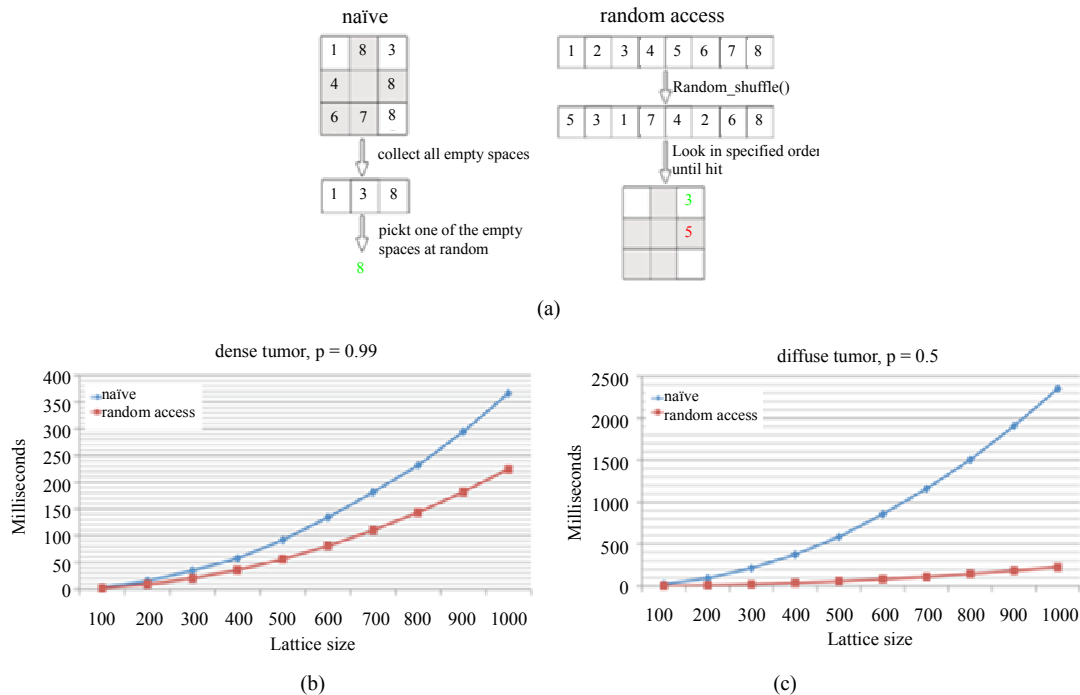

Figure 4. Comparison of two different procedures to select a random free spot from cell neighborhood. Free spot is selected iteratively for each cell on the lattice. (a) Naïve procedure visits all neighboring spots, temporarily stores those that are vacant and chooses a random element from the temporary vector. Random access procedure uses a predefined vector of neighboring lattice sites that is randomly shuffled. The cell neighborhood is searched in that random order and then returns the first vacant site; (b) and (c) Average evaluation times of different lattice sizes for dense tumors (b) and diffusive tumors (c).

Dependent on the modeled cellular processes, additional alterations or improvements may be required. For example, one may choose to store hashed information about the cell neighborhood in the lattice. In particular, a limited number of possible cell neighborhood configurations may be encoded in identifying keys.

3.4. Random Ordering

Many programming languages provide efficient procedures and data structures that can be utilized for cellular automata design in combination with simulation specific code. In asynchronous stochastic cellular automata of tumor growth random cell ordering and random access of cells is fundamental. A naïve implementation of selecting cells in a random order may consist of

1) From a vector containing all cells, pick a cell at random by drawing a random positive integer not larger than the vector length;

2) Erase the selected cell from the vector to avoid its reselecting;

3) Repeat steps 1 and 2 until there are no cells left in the vector.

The C++ Standard Template Library (STL) provides numerous algorithms to perform search, sort and shuffle operations. Random shuffle rearranges all elements in a specified range randomly in a single invocation. The STL random shuffle procedure reduces computation time compared to the naïve approach by multiple orders of magnitude even for small vector sizes (**Figure 5**), clearly demonstrating the power and importance of using standard language-specific data structures and algorithms for high-performance simulations.

3.5. Dynamically Growing Domains

To simulate a growing tumor population from a single cancer cell computational lattice-induced boundary constraints need to be avoided. An appropriate lattice size must be selected dependent on the achievable tumor size, which requires a priori knowledge about emerging population dynamics, tumor density and cell diffusibility. A dense, radially symmetrically growing two-dimensional tumor population of 100,000 cells could well fit into a 400×400 lattice. Cells in a highly diffusive irregular tumor, however, will likely hit the boundary of such lattice early during tumor growth. Whilst a sufficiently large lattice could ensure avoidance of boundary contact, memory requirements and computing performance limit such approach. Large amounts of computational resources would be wasted especially early in population expansion when only a few cells are present.

One possibility is to use dynamic data structures such as a C++ standard template library (STL) map, which can be understood as an associative container that stores elements formed by the combination of a key value and a mapped value. Unfortunately, accessing elements in a map is logarithmic in size, which for large tumor sizes dramatically decreases computational performance (**Figure 6**). We propose a dynamically allocated array with associated procedures that expand the lattice upon cell boundary contact by a fixed amount of lattice points. While a static large array of 1000×1000 lattice points is the most efficient for large tumors it is inefficient for small tumors up to 10,000 cells due to unoccupied lattice sites occupying large amounts of memory. The dynamically expanding array is most efficient for small tumors and comparable to large static arrays for large tumor populations.
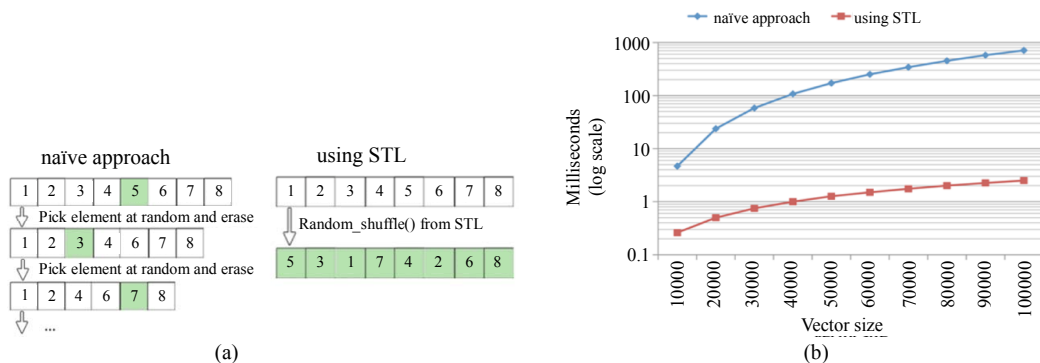

Figure 5. Comparison of the efficiency of the built-in STL shuffling procedure and the procedure coded in the naïve way. (a) Naive procedure iterates the steps consisted of picking random element from the vector and then erasing it until the initial vector is empty. Single invocation to the STL procedure gives us the shuffled vector; (b) Evaluation times for those two procedures for different sizes of the shuffled vector.
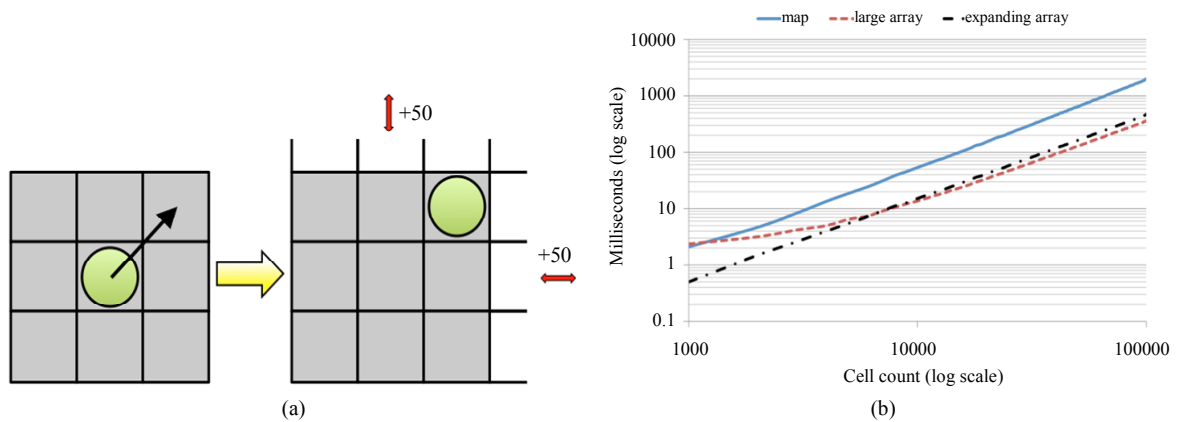
Figure 6. Dynamically growing domain. (a) Computational lattice expands by 50 grid points in the direction(s) where a cells reaches the boundary; **(b)** Simulations of simple tumor growth from a single cell when each cell (cells are considered in linear order) proliferates into a random free adjacent spot until the tumor population reaches the predefined final size (no other action is considered). Evaluation of computing time to reach different cell counts for three types of domains: STL map (solid blue curve), large array (red dashed) and dynamically expending array (black dash-dotted).

3.6. Tumor Growth Simulations

We will use a combination of the presented high-performance techniques to simulate solid tumor growth and compare it to a naïve implementation. Let us initialize tumor growth simulations with one cancer stem cell located in the center of a square lattice with trait vector [cct = 24 hours, $\rho = \infty$, $\mu = 100$ μm/day, $\alpha = 0\%$] and $p_s = 0.1$. Non-stem daughter cells are initialized with trait vector [cct = 24 hours, $\rho = 10$, $\mu = 100$ μm/day, $\alpha = 1\%$].

These parameter values have previously been shown to enable fast dense tumor growth [20]. Tumor growth dynamics with other parameters have been discussed elsewhere [22-24]. Cell cycle time and migration rate can be measured experimentally [17], while other parameters are yet to determined. It is conceivable, however, that model parameters are organ or even patient specific. We therefore use the initial trait vector as control and study the sensitivity of tumor volume to changes in each parameter. We simulate tumor growth for t = 180 days using an intuitive implementation (naïve code) and compare to an implementation with a combination of above discussed improvements (improved code). The naïve simulation is executed on a fixed 750×750 square lattice, whereas the improved simulation is initiated on a 50×50 square lattice with dynamically expanding domains. Due to the stochastic nature of the model we simulate N = 100 (improved) and N = 77 (naïve) independent tumors and report average results. Both implementations yield similar population sizes with comparable cancer stem cell and non-stem cancer cell numbers (**Figure 7**). While the naïve code executes in an average of 4212 seconds (>70 minutes) the improved code executes in 51 seconds (<1 minute)—an 82-fold reduction in computing time. The high-performance of the improved code is due to the dynamically expanding domain as well as efficient access to information on vacant neighboring lattice sites. More than 70% of all cells at the final time point of the simulation have no adjacent space, and less than 5% of cells have two or more vacant lattice sites to migrate or proliferate into (**Figure 7(d)**). Graphical visualization of tumor morphologies at different time points show that tumors simulated with either implementation technique are non-differentiable beyond intrinsic stochastic effects (**Figure 8**).

To study the effects of model parameters on emerging tumor volume we separately change the initial parameter values (control) by 50% in either direction and compare resulting total cell count to the control tumor. Sensitivity analysis reveals that spontaneous rate of cell death has little impact on total cell number, compared to cell migration and symmetric stem cell division rate that strongly correlate with tumor volume. Whilst an increase in symmetric stem cell division probability p_s and cell migration rate μ yields larger tumor volumes, cell death rate α and proliferation potential ρ modulate tumor volume non-monotonically (**Figure 9**). A 50% parameter value increase as well as decrease yields tumor volumes that are smaller than the control tumor, suggesting the existence of optimum parameter values that maximize tumor volume. Their specific values, however, will be dependent on the other model parameters [23]. Interestingly, increasing proliferation potential ρ yields significantly smaller tumors than lower ρ values, indicating a strong competition between non stem cancer cells and cancer stem cells that drives the total population into prolonged phases of dormancy [24].
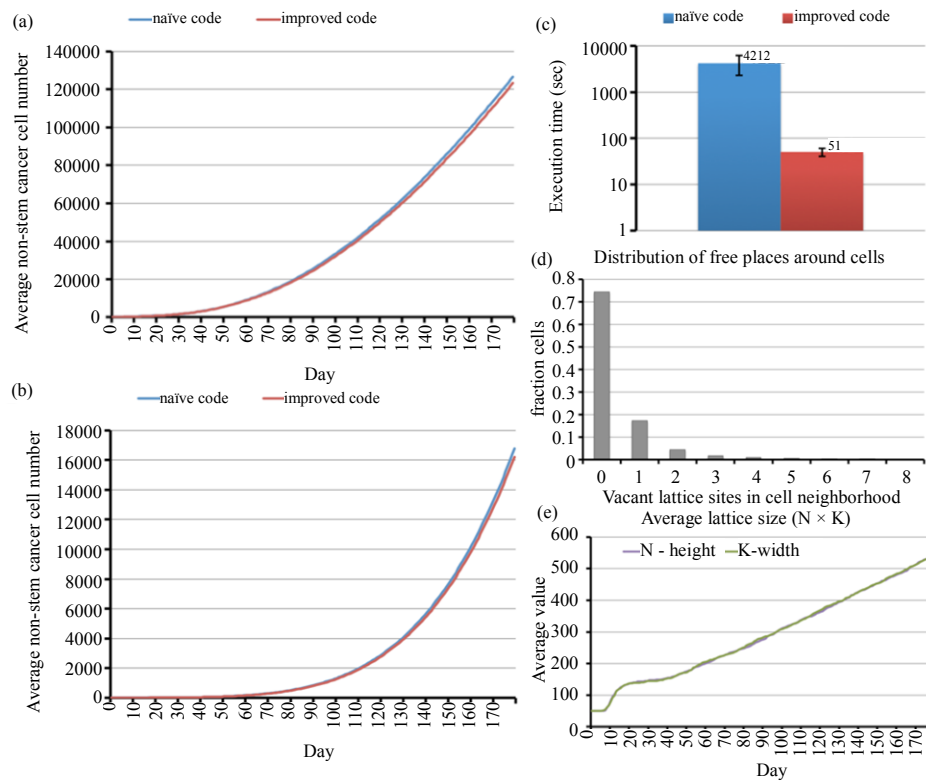
Figure 7. Model results for simulating 180 days of tumor growth initiated by one cancer stem cell. Naïve (blue curve) and improved (red) code simulate comparable tumor growth dynamics with similar non-stem cancer cell (a) and cancer stem cell (b) numbers; (c) Average simulation times and standard deviations; (d) Distribution of vacant lattice sites in cell neighborhood at final simulation time point; (e) Evolution of average lattice size in the improved code. Initial lattice size 50 × 50, final lattice size 550 × 550. Shown are averages (standard deviations ommited in most panels for clarity) for N = 100 (improved code) and N = 77 (naïve) independent simulations.

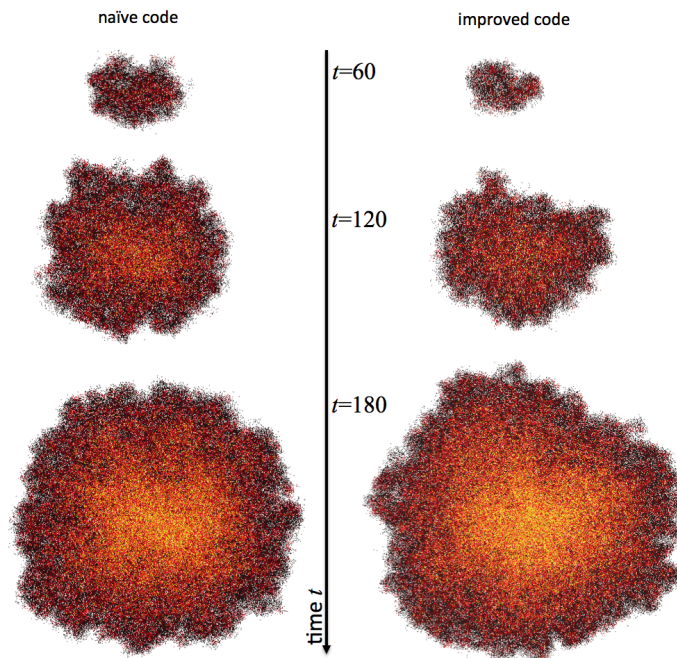
Figure 8. Representative tumor morphologies simulated with naïve (left) and improved (right) code. Colors as defined in Figure 1(a).
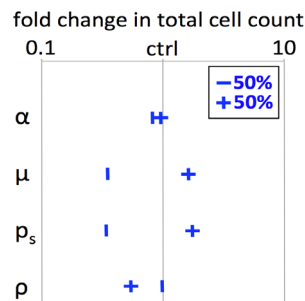
AM

Figure 9. Cancer stem cell driven tumor growth model parameter sensitivity analysis. All initial model parameters (control, ctrl) were changed −50% or +50% and average tumor volumes after t = 180 days (N = 50) compared to the control volume (log scale). Error bars ommitted for clarity.

4. Discussion

Cellular automata are frequently used to simulate solid tumor growth and cancer stem cell dynamics [1,25,26]. Intuitive design of stochastic cellular automata for tumor modeling is often counterbalanced by its performance. Although cellular automata are lattice-based, naïve implementation as two-dimensional *Boolean* arrays has little computational efficiency. We set out to compare C++ data structures, memory-efficient procedures and dynamic domains to decrease computing time. As extension to three spatial dimensions is trivial, we presented implementation details in two dimensions for clarity. We found that simple substitutions in intuitive cellular automaton implementations significantly decrease computing time. First, appropriate use of data type *char* over *int* provides a 4-fold reduction in memory allocation. Second, consideration of a coded lattice that holds information about a cell's neighborhood vacancies rather then *Boolean* information whether or not a cell is occupying that lattice point significantly decreases computation time if queries about adjacent space are frequently required for cell decisions. Third, utilization of the C++ STL shuffle method to provide a random order of elements proves superior to repeatedly selecting single elements at random positions within a vector. Finally, we presented a dynamically growing domain that evolves according to the population size, which keeps compuation time exceptionally low when the population is small. Computing time is comparable to a large array when the population approaches array carrying capacity, but with the unique option to further expand the domain when needed. When all of these adjustments are combined into a simulation of cancer stem cell driven solid tumor growth, the improved implementation yields a high-performance over the naïve approach. In simulations of tumor growth for 180 days from a single cell to a population of about 140,000 cells, the presented high-performance cellular automaton yields an 82-fold reduction in computing time while reproducing the results of the naïve implementation. We believe the developed high-performance cellular automaton will serve as a template for future simulations of solid tumor growth as well as other population dynamics models. We share the source code for the presented naïve and improved code on our personal websites and the sourceforge.net repository.

Acknowledgements

REFERENCES

[1]   Z. Agur, Y. Daniel and Y. Ginosar, "The Universal Properties of Stem Cells as Pinpointed by a Simple Discrete Model," *Journal of Mathematical Biology*, Vol. 44, No. 1, 2002, pp. 79-86. http://dx.doi.org/10.1007/s002850100115

[2]   A. R. A. Anderson, "A Hybrid Mathematical Model of Solid Tumour Invasion: The Importance of Cell Adhesion," *Mathematical Medicine and Biology*, Vol. 22, No. 2, 2005, pp. 163-186. http://dx.doi.org/10.1093/imammb/dqi005

[3]   P. Gerlee and A. R. A. Anderson, "An Evolutionary Hybrid Cellular Automaton Model of Solid Tumour Growth," *Journal of Theoretical Biology*, Vol. 246, No. 4, 2007 pp. 583-603. http://dx.doi.org/10.1016/j.jtbi.2007.01.027

[4]   A. Bankhead III, N. S. Magnuson and R. B. Heckendorn, "Cellular Automaton Simulation Examining Progenitor Hierarchy Structure Effects on Mammary Ductal Carcinoma *in Situ*," *Journal of Theoretical Biology*, Vol. 246, No. 3, 2007, pp. 491-498. http://dx.doi.org/10.1016/j.jtbi.2007.01.011

[5]   H. Hatzikirou, L. Brusch, C. Schaller, M. Simon and A. Deutsch, "Prediction of Traveling Front Behavior in a Lattice-Gas Cellular Automaton Model for Tumor Invasion," *Computers and Mathematics with Applications*, 2009, pp. 1-14.

[6]   A. R. Kansal, S. Torquato, G. R. Harsh IV, E. A. Chiocca and T. S. Deisboeck, "Cellular Automaton of Idealized Brain Tumor Growth Dynamics," *BioSystems*, Vol. 55, No. 1-3, 2000, pp. 119-127. http://dx.doi.org/10.1016/S0303-2647(99)00089-1

[7]   J. Tang, H. Enderling, S. Becker-Weimann, C. Pham, A. Polyzos, C.-Y. Chen, *et al*., "Phenotypic Transition Maps of 3D Breast Acini Obtained by Imaging-Guided Agent-Based Modeling," *Integrative Biology*, Vol. 3, No. 4, 2011, pp. 408-421. http://dx.doi.org/10.1039/c0ib00092b

[8]   T. Alarcón, H. M. Byrne and P. K. Maini, "A Cellular Automaton Model for Tumour Growth in Inhomogeneous Environ-ment," *Journal of Theoretical Biology*, Vol. 225, No. 2, 2003, pp. 257-274. http://dx.doi.org/10.1016/S0022-5193(03)00244-3

[9]   A. A. Patel, E. T. Gawlinski, S. K. Lemieux and R. A. Gatenby, "A Cellular Automaton Model of Early Tumor Growth and Invasion: The Effects of Native Tissue Vascularity and Increased Anaerobic Tumor Metabolism," *Journal of Theoretical Biol-ogy*, Vol. 213, No. 3, 2001, pp. 315-331. http://dx.doi.org/10.1006/jtbi.2001.2385

[10] B. Ribba, T. Alarcón, K. Marron, P. K. Maini and Z. Agur, "The Use of Hybrid Cellular Automaton Models for Improving Cancer Therapy," 2004, pp. 444-453.

[11] M. Aubert, M. Badoual, S. Féreol, C. Christov and B. Grammaticos, "A Cellular Automaton Model for the Migration of Gli-oma Cells," *Physical Biology*, Vol. 3, No. 2, 2006, pp. 93-100. http://dx.doi.org/10.1088/1478-3975/3/2/001

[12] M. J. Piotrowska and S. D. Angus, "A Quantitative Cellular Automaton Model of *in Vitro* Multicellular Spheroid Tumour Growth," *Journal of Theoretical Biology*, Vol. 258, No. 2, 2009, pp. 165-178. http://dx.doi.org/10.1016/j.jtbi.2009.02.008

[13] Y. Jiao and S. Torquato, "Emergent Behaviors from a Cellular Automaton Model for Invasive Tumor Growth in Heterogeneous Microenvironments," *PLOS Computational Biology*, Vol. 7, 2011, Article ID: e1002314. http://dx.doi.org/10.1371/journal.pcbi.1002314

[14] G. G. Powathil, K. E. Gordon, L. A. Hill and M. A. J. Chaplain, "Modelling the Effects of Cell-Cycle Heterogeneity on the Response of a Solid Tumour to Chemotherapy: Biological Insights from a Hybrid Multiscale Cellular Automaton Model," *Journal of Theoretical Biology*, Vol. 308, 2012, pp. 1-19. http://dx.doi.org/10.1016/j.jtbi.2012.05.015

[15] H. Enderling, D. Park, L. Hlatky and P. Hahnfeldt, "The Importance of Spatial Distribution of Stemness and Proliferation State in Determining Tumor Radioresponse," *Mathematical Modelling of Natural Phenomena*, Vol. 4, No. 3, 2009, pp. 117-133. http://dx.doi.org/10.1051/mmnp/20094305

[16] M. Gardner, "Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game 'life'," *Scientific American*, Vol. 223, No. 4, 1970, pp. 120-123. http://dx.doi.org/10.1038/scientificamerican1070-120

[17] X. Gao, J. T. McDonald, L. Hlatky and H. Enderling, "Acute and Fractionated Irradiation Differentially Modulate Glioma Stem Cell Division Kinetics," *Cancer Research*, Vol. 73, No. 2, 2013, pp. 1481-1490. http://dx.doi.org/10.1158/0008-5472.CAN-12-3429

[18] J. Tang, I. Fernandez-Garcia, S. Vijayakumar, H. Martinez-Ruiz, I. Illa-Bochaca, D. H. Nguyen, *et al.*, "Irradiation of Juvenile, but Not Adult, Mammary Gland Increases Stem Cell Self-Renewal and Estrogen Receptor Negative Tumors," *Stem Cells*, 2013. http://dx.doi.org/10.1002/stem.1533

[19] Trisilowati and D. G. Mallet, "Experimental Modeling of Cancer Treatment," *ISRN Oncology*, Vol. 2012, 2012, Article ID 828701. http://dx.doi.org/10.5402/2012/828701

[20] H. Enderling, A. R. A. Anderson, M. A. J. Chaplain, A. Beheshti, L. Hlatky and P. Hahnfeldt, "Paradoxical Dependencies of Tumor Dormancy and Progression on Basic Cell Kinetics," *Cancer Research*, Vol. 69, No. 22, 2009 pp. 8814-8821. http://dx.doi.org/10.1158/0008-5472.CAN-09-2115

[21] J. L. Hennessy, D. A. Patterson, "Computer Architecture," Elsevier, Amsterdam, 2012.

[22] H. Enderling, L. Hlatky and P. Hahnfeldt, "Migration Rules: Tumours Are Conglomerates of Self-Metastases," *British Journal of Cancer*, Vol. 100, No. 12, 2009, pp. 1917-1925.

[23] C. I. Morton, L. Hlatky, P. Hahnfeldt and H. Enderling, "Non-Stem Cancer Cell Kinetics Modulate Solid Tumor Progression," *Theoretical Biology and Medical Modelling*, Vol. 8, 2011, p. 48. http://dx.doi.org/10.1186/1742-4682-8-48

[24] H. Enderling, "Cancer Stem Cells and Tumor Dormancy," In: *Advances in Experimental Medicine and Biology*, Springer, New York, 2012, pp. 55-71.

[25] H. Enderling, L. Hlatky and P. Hahnfeldt, "Cancer Stem Cells: A Minor Cancer Subpopulation That Redefines Global Cancer Features," *Frontiers in Oncology*, Vol. 3, 2013, p. 76. http://dx.doi.org/10.3389/fonc.2013.00076

[26] A. Sottoriva, J. J. C. Verhoeff, T. Borovski, S. K. McWeeney, L. Naumov, J. P. Medema, *et al*., "Cancer Stem Cell Tumor Model Reveals Invasive Morphology and Increased Phenotypical Heterogeneity," *Cancer Research*, Vol. 70, No. 1, 2010, pp. 46-56. http://dx.doi.org/10.1158/0008-5472.CAN-09-3663