CellCom:
A Hybrid Cellular Automaton Model of Tumorous Tissue Formation and Growth

David Rodrigues[1], Jorge Louçã[2]

[1] MSc Student in Complexity Sciences, ISCTE – Instituto Superior de Ciências do Trabalho e da Empresa. Av.ª das Forças Armadas – 1649-026 Lisboa, Portugal
{m4467@iscte.pt}
[2] Supervising Professor, ISCTE – Instituto Superior de Ciências do Trabalho e da Empresa. Av.ª das Forças Armadas – 1649-026 Lisboa, Portugal
{Jorge.L@iscte.pt}

Abstract. We propose a descriptive 2D Hybrid Cellular Automaton model that simulates the growth behavior of a tissue with cancerous cells. We combine a set of hallmarks of cancer (Hanahan and Weinber, 2000) with the effect of acidification of the cell surroundings by the production of acid by tumorous cells and model some aspects of the communication in living tissues. The dynamics of the growth is determined to obey a power law. We conclude that the model predicts a cancer growth rate between the exponential Gompertzian growth and a laboratory evidence of linear growth.

Keywords: Simulation, Cancer Growth, Cellular Automaton, Hallmarks of Cancer.

1 Introduction

The stronger you are you get to see more connections. The pieces and positions come alive with patterns. You start seeing possibilities in these patterns. (…) As you refine your analysis the more sharp your understanding of the position gets. The pieces are there standing and suddenly when you see the connections they become beautiful.
Vishy Anand, (Chess Player, n.1 in FIDE rank); April, 24th 2007.

There was an estimated 6,7 million people deaths in 2003 from cancer worldwide and 10,9 million more were diagnosed (Parkin et al., 2005). This numbers are expected to grow with time as populations get older in developing countries and better cancer treatments are promoting longer longevity. Until now cancer treatment has been generally limited to a specific form of the disease. Most cancer research has been focused in particular reactions and signal transduction pathways. This offer ways for therapy, but are specific to that form of cancer. At this level of detail each variety of cancer is unique and a complicated phenomenon.

The cancer research is one field where most money is spent in today's research. The understanding of the dynamics of formation and cancer growth can give researchers opportunities to try new prevention and treatment solutions. Computer simulations have been to used to study *in silico* some of the process that are thought to lead to the formation of cancerous masses and have attained good results in explaining some facts observed *in vivo* and *in vitro* tumors.

We want to take an approach that can describe the problem in a meaningful way, but avoiding the caveats of going into a very demanding computational and reductionist approach. A reductionist approach would lead the simulation to a scale of molecular dynamics where there's still much uncertainty how intra-cellular dynamics evolve. On the other way a holistic view of this problem would explain the global dynamics of cancer formation and growth in terms of some macroscale properties but would fail in pinpointing the micro causes.

We believe that somewhere in the middle of this two views is possible to create a framework that is still computational feasible and that would help explain both some of the macroscale dynamics has some microscale mechanisms.

This kind of framework would allow itself to be extended in the future in both directions, up and bottom, by the inclusion of refined submodels for those aspects of interest.

In this work we are interested in answering some questions that might help understand this problematic. We defined a research hypothesis where we state that it is possible to use simulation tools in which we could integrate several levels of abstraction of the reality in a way that would allow us to have at least a qualitatively description of the dynamics of the process. We are interested in the level of communication between cells and how this chemical signaling can be conducted *in silico*. Another level of interest is that of the gene expression and eventually the mechanics of it. And in a third level we are interested in the macroscale analysis of the patterns that emerge in the cancerous tissue.

This contribution demonstrates the possibility of using computer simulations to attain a descriptive model of the cancer formation and growth using very simple rules.

The main body of this work is divided in three main blocks. We'll start with a bibliographic review of some aspects that are pertinent to the simulation of biological systems and in particular to the simulation of cell tumors. Then we'll introduce our proposed model, CellCom, with a descriptive methodology proposed by Grimm followed by the presentation of the experiments and results obtained. Finally we discuss the model presented and also focus on some topics that would be interesting to develop in further research.

2 Bibliographic review.

Overview

Cellular systems and subsystems can be simulated at three different scales: the nanoscale, the mesoscale and the continuum or macroscale. At the level of the atoms and molecules we typically use molecular dynamics to model the behavior of 100's to 1000's of discrete atoms over relatively short periods of time (10^{-10} to 10^{-9} s) and space (10^{-9}m). Molecular dynamics methods, which treat the atoms and bonds in a semi-classical manner, are fully deterministic and remarkably accurate over the short temporal and spatial scales that are normally simulated. This limits the number of molecules that can be simulated and therefore when it is needed to model at a macroscale we need to turn to ordinary (ODE) or partial (PDE) differential equations. This makes the model a continuum where molecules essential loose their discreteness and became infinitely small and numerous (Wishart 2004). But even at this continuum level it isn't possible to describe every system in terms of differential equations, and not all are solvable. This is where a middle approach is needed. One that still captures the discrete aspects of molecules but allows for an upper analysis and modeling. This is where simulation incorporating different abstraction levels can help connecting the nanoscale and macroscale in the mesoscale of modeling.

In order to understand the field of research it is necessary to review some aspects that need to be taken into account. Mainly we will address in the following sections the description of the cellular life-cycle, we'll characterize some aspects of tumorous masses and it's growth dynamics, we'll refer other aspects like the fractality of the contour of growing tumors and the use of nanomachines that simulate molecular communication. Also we'll end this section by reviewing some software programs and models that were developed to explain some aspects of this field of research. We'll start by describing succinctly the cell life-cycle.

Cell Life-Cycle

The first step to understand the problem of tumor growth is look at the cell life cycle.

A cell in an adult organism can be viewed as a steady-state system. The DNA is continuously read into mRNAs, which allow the production of proteins. As the proteins function they are also being degraded and replaced by new ones and the system is balanced and the cell neither grows, shrinks or changes its function. This static overview of the cell doesn't give insights to its life cycle dynamic aspects

The dynamics of a cell can best be understood by examining the course of its life. Each cell arises from the division of a parent cell into two daughter cells. The sequence of events that lead to this division is called the cell-cycle and has a major importance as it is the mechanism by which any type of cell grows and multiplies.

Basically, the cell-cycle is composed of 4 steps in sequence that act as an internal clock for the cell life (Alberts et al. 1998):

G1 phase – in this phase the cell checks to see if there are conditions to initiate DNA replication. It acts has a checkpoint to see if the cell has the size and environmental conditions to continue with the division process. In this phase the cell also verifies if the DNA is damaged. If the cell verifies that all conditions are met, then it will trigger the next phase where DNA multiplication occurs.

S phase – in this phase the cell will start replicating it's DNA. It is also called the synthesis phase and at it's end will have a double stranded DNA.

G2 phase – This phase acts as a checkpoint similar to G1. The cells checks to see if the DNA replication has ended correctly and checks if it's size and environment of the cell allow it to enter the mitosis phase were actual division occurs. As in the G1 phase, the cell has mechanisms to halt the progress of cellular division at this point waiting for the conditions to continue the process. These three phases are usually grouped and called the interphase.

M phase – this phase is were the division of the cell occurs. The mitosis in itself is sequence of steps through which the division takes place. From the condensation of the chromosomes and formation of the mitotic spindles that will pull each pair of chromosomes to its spindle pole, to the division of the cytoplasm creating the two daughter cells, all occur in the M phase.

It is a general rule that mammalian cells will multiply only if they are simulated to do so by signals from other cells. If deprived of such signals, the cell cycle arrests at a G1 checkpoint and enters the G0 state. G0 is a modified G1 state in which the cell-cycle control system is partly dismantled (Alberts et al. 1998).

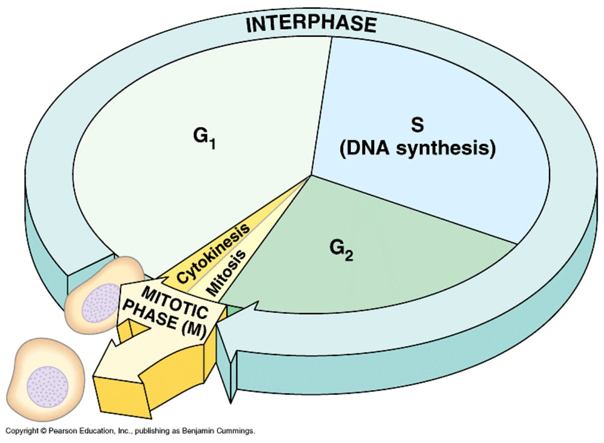

Figure 1 – Cell Life Cycle

Cells spend their life in one of two states depending if they are allowed to replicate or not. If they are in the quiescent state, they don't reproduce and this state called G0. If they are allowed to replicate then they enter the G1 phase and will proceed through their cell-cycle up to the point where two daughter cells are formed from one parent cell.

Tumor Growth

How cancer grows and how it is initiated has been one field of great study by the medical community. We wont to go over an exhaustive analysis of what has been accomplished in recent years as it would reveal a daunting task as the number of publications and researches in this area is overwhelming. We will review some of the papers that we studied during the preparation for the construction of our model. This will give an insight of what is being done in tumor analysis and will be the grounds on which we made our proposition.

Hallmarks of Cancer

Hanahan and Weinberg (Hanahan and Weinberg, 2007) wrote a paper called "The Hallmarks of Cancer" where they stated that tumorigenesis in humans is a multistep process and that those steps reflect genetic alterations that drive the progressive transformation of normal human cells into highly malignant derivatives. From the many cancers diagnosed in humans they say that there's an age-dependent incidence implicating four to seven rate-limiting, stochastic events. They state that "taken together, observations of human cancers and animal models argue that tumor development proceeds via a process formally analogous to Darwinian evolution, in which a succession of genetic changes, each conferring one or another type of growth advantage, leads to the progressive conversion of human cells into cancer cells".

They proposed that only six cellular alterations are essential to malignant growth. These six hallmarks are believed to be common to most human tumors. The phenotypic changes at the cellular level that are essential hallmarks are: unlimited mitosis, ignoring growth-inhibition signals, escaping dependence on external growth stimulation, the ability to recruit new vascular structures, modality and invasion and disabling the mechanisms that normally detect mutation and trigger apoptosis. To this six genetic instability is added has a factor that accounts for the high incidence of mutations on cancer cells (Abott et al., 2006).

From the vast catalog of cancer cell genome types the authors proposed that the manifestation of six essential alterations in cell physiology will dictate malignant growth. These hallmarks are resumed in the following list:

1) self-sufficiency in growth signals

2) insensitivity to growth-inhibitory (anti-growth) signals
3) evasion to programmed cell death (apoptosis)
4) limitless replicative potential
5) sustained angiogenesis
6) tissue invasion and metastasis

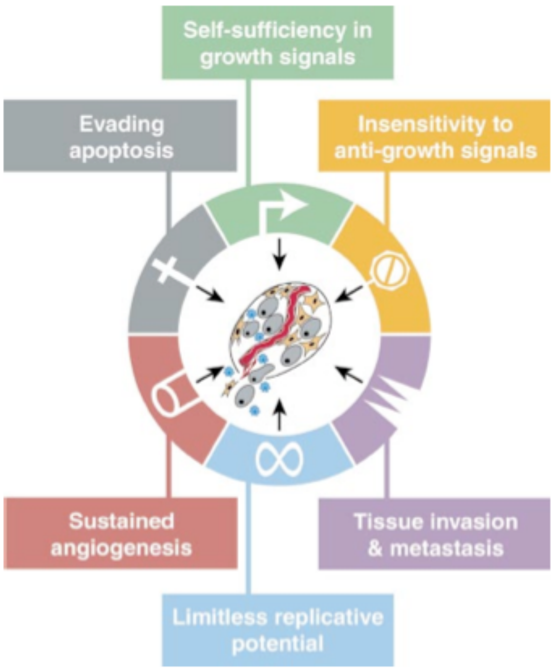
Self-sufficiency in growth signals
Evading apoptosis
Insensitivity to anti-growth signals
Sustained angiogenesis
Tissue invasion & metastasis
Limitless replicative potential

Figure 2 – Hallmarks of Cancer

The authors also refer that the acquisition of this six capabilities during the course of tumor progression creates a dilemma as while evidence suggests that most of them are acquired directly or indirectly, through changes in cancer cells, the monitoring and repairing mechanisms in normal cells make this mutations rare. The genome maintenance system strive to ensure that DNA sequence information remains pristine, and the checking mechanisms would make tumor cell genomes highly unlikely to occur within the human life span. Yet, tumors do appear at a substantial frequency, making several authors state that tumor cells must acquire increased mutability in order for the process of tumor growth to be successful. The authors refer that from those systems capable of increasing cell mutability, the most prominent is the p53 tumor suppressor protein, which in response to DNA damage, elicits either cell cycle arrest to allow DNA repair to take place or apoptosis if the damage is excessive. The authors say that is clear that the p53 signaling pathway is lost in most human cancers.

This makes the hallmarks of cancer a set of seven instead of six manifestations, although six of might be considered a phenotypic manifestation of genome mutations and the seventh a multiplication factor that will increase the rate at which the first six will occur.

Fractality and Growth Laws

One interesting aspect of the tumor dynamics that has been studied by Brú et al. in 2003, is that cell colonies are fractal, and a classical Euclidean geometry description of the growth contours is very difficult to provide. These authors did a fractal analysis of the nature colonies of 15 lines of cancer growing *in vitro* and as well 16 others growing *in vivo* and they concluded that all colonies had the same growth dynamics, which corresponds to the molecular beam epitaxy (MBE) universality class. They also calculated characteristic fractal dimensions for all of the lines of cancer.

This meaning that the cell aggregates are characterized by 1) a linear growth rate in the sense that the term “linear” means that the colony radius grows linear with time, 2) the constraint of growth activity to the outer border of the cell colony or tumor and 3) diffusion at the colony surface of cells. This evidence is in contradiction with other models of cancer growth that state that the dynamics is characterized by a Gompertzian growth.

Molecular Communication

There's been some recent research in this field with its interdisciplinary scope ranging from nanotechnology, biotechnology to communication technology. Molecular communication is inspired by the observation that in biological systems, communication is typically done through molecules. Hiama et al. have studied molecular Communication in the attempt to apply these concepts to nanomachines communication. These machines are molecular scale objects that are capable of performing simple tasks such as actuation and sensing. According to Whitesides (Whitesides, 2001) there's two types of nanomachines: artificially created machines that mimic traditional machines, and nature made nanomachines, also called soft nanomachines which are found in biological systems.

The most interesting aspect of the use of the nanomachines research in the context of molecular communication is that the authors defined a set of steps that must be present to make communication flow from a sender to a receiver. The proposed mechanisms might be considered as a submodel to implement some aspects of the virtual cell.

Software Simulations

We will now review some software packages that are used in cellular simulation and cancer growth modeling.

SymCell (Wishart et al, 2004)

This program is a dynamic cellular automaton. This means basically that agents are placed in one position and that they can assume a state from a predefined set of states and then change states according to a set of rules. Usually these rules represent some sort of environmental representation of the agent's surroundings. The dynamic cellular automaton is one where its agents are not confined to a single spot in the lattice of those more traditional CAs, but can move around the lattice to simulate Brownian movement, diffusion, convection, or any other property that might be appropriate for the agent to have.

Wishart *et al.*, used this approach in their SymCell software, where the user can use it to simulate cellular and biochemical processes, through a DCA (dynamic cellular automaton) algorithm. They provide a simple user interface allowing the user to drag & drop elements into the simulation for which, the user must then introduce some parameters.

The software is capable of implement some of the cellular characteristics:

a) Small molecules.
b) Membrane.
c) Membrane proteins
d) DNA molecules
e) Genes

The software allows the user to create interactions between these components in ways that mimic the behavior of real cells.

After the creation of the components, the simulator will model the process and simulate it. Each step of the simulation represents 1 ms and the movable elements placed by the user can move in theirs Moore neighborhood.

Also the simulator can use data in the Systems Biology Markup Language (SBML) witch allows the user to import data from available databases.

This system has its virtues but also is problems. Modeling at a macroscale (at least at cellular level) implies that the discreet approach of the DCA doesn't allow matching the atomic spatial complexity of something like an enzyme. Also these models can't predict certain properties of the system, like molecular properties, and for those other method should be used. Those results can than be included in the DCA, but not the other way around.

CancerSim (Abott et al. 2006)

CancerSim implements the hallmarks of cancer described by Hanahan and Weinberg in the *Cell* magazine in 2000. The simulation consists of cells and a circulatory system, both of which grow according to their own rules. The model is built as a 3D cellular automaton where each place (cube) contains either one cell or an empty space, but where the vascular system may pass through that cube without restrictions. CancerSim is implemented in C++ and available under a free license and exist in several flavors of Unix, MacOS and Windows.

In this model, the authors implemented the Hallmarks of cancer: self-sufficiency in growth signals, sustained angiogenesis, insensitivity to growth inhibitor signals, evasion to apoptosis, limitless replicative potential and genetic instability. They didn't implement tissue invasion and metastasis. Although genetic instability isn't considered one hallmark of cancer, evidence showed that in normal conditions cells wouldn't have the many chances to convert to cancer cells during the life-span of human life. The authors state this genomic instability must therefore be present (Hanahan et al. 2000). The genomic instability is modeled in CancerSim as a switch in the pseudo-genome of the cell that will have a multiplying impact on the probability of occurrence of other mutations.

Patel et al. 2001

Patel et al. (Patel et al., 2001) have proposed a cellular automaton model of the early tumor growth and invasion. Their model proposes a hybrid cellular automaton that incorporates normal cells, tumor cells, necrotic space or empty space and a random network of native microvessels as the components of the state vector of each square in the lattice of the CA. The authors use a set of differential equations to model the diffusion of H^+ and Glucose, being the former largely resulting from the tumor's excessive reliance on anaerobic metabolism. In their paper they showed that high H+ ion formation is favorable to tumor production (but not to normal cells). However, for each pH there's an optimal microvessel density for which growth and invasion is most successful. This leads to a local optimal concentration of acid for the tumor, but not for the normal cells.

The model they presented is a NxN lattice of automaton elements each with a vector state and a rule-set governing their evolution. Each lattice element regardless of its state and occupation enforces a correspondence between automaton elements and physical cells comprising the tissue. This elements have a physical size of $\Delta=20\mu$m. The simulation is carried by setting up a random network of vascular cells in the model and then placing a group of 5 cells in the middle of the lattice. The simulation then performs according to a simple set of rules:

a) If the automaton element is vacant or occupied by a micro vessel then he wont evolve directly, although in the case of the former it can evolve indirectly by he division of another cell.
b) If the occupancy of the automaton is either a tumor or a normal cell then the concentrations of H^+ and Glucose are considered and pH thresholds for each type of cell decide the evolution of this automaton element.
c) If the cell survives because pH is above its threshold then it will be given opportunity do divide. Each cell will only survive in this step if the concentration of Glucose is above a certain value, and the division will only be successful if there's a vacant cell in the vicinity.
d) The remaining values of the state vector that are described by continuous properties are then updated by the solving of two differential equations.

This model was written in C with simulations of lattices 100x100 and 200x200 (for selected parameters) and run for 40 generations on a DEC Alpha Unix workstation.

The authors conclude that the H^+ production by tumor cells in the early tumorigenesis, when only a few tumor cells exist, would be sufficient to significantly alter the environment, although the model showed that from a small number of tumor cells, the mass will develop into a clinically important malignancy if the clonal phenotype alters the local microenvironment so that it is hostile to normal cells. They affirm that this depends on the balance of the acid produced and the acid removed by the local blood flow. In this approach they examine the influence of vascularity and conclude that for each rate of acid production there is an optimal density of microvessels that facilitate the removal of excess acid in the system.

Kansal et al. 2000

"Simulated Brain Tumor Growth Dynamics Using a Three-dimensional Cellular Automaton"

Kansal et al. (Kansal et.al, 2000) developed a three-dimensional model of brain cancer growth that using a small set of parameters show macroscopic behavior identical to those of real tumors, mainly the Gompertzian growth for cancers growing nearly three orders of magnitude in radius. Their model also predicts the composition and dynamics of the tumor at selected time points in agreement with medical literature. This model presents some features that are worth mentioning:

a) The ability of cells to divide is treated by redefining the transition between dividing and non-dividing cells, as the cells attempt to divide, they will search for sufficient space for the new cell, beginning with its neighbors and expanding outwards until they find an empty cell or nothing is found within the proliferation radius. If the cell attempts to divide but cannot find space it is turned into a non-proliferative cell
b) The CA used is modeled in a 3D lattice constituted by Voronoi tessellation, and is isotropic in space avoiding the creation of artificial anisotropies possible with square or cubic lattices. The Voronoi lattice used in the model defined neighbors of a cell by those who share a common face.
c) The lattice has a varying density sites (adaptive grid lattice) that allows small tumors to be simulated with greater accuracy an still allowing them to grow large in size (three orders of magnitude)

This model allowed the authors to simulate the growth of small populations of about 1000 real cells to a fully developed tumor with 10^{11} cells. This number of cells requires great computational power and the simulations were run in an IBM SP2 Parallel computer.

This model idealized a tumor with as a spherical body consisting of several concentric shells. The inner core was composed of necrotic cells, which radius was a function of time. The next shell contained cells that where alive but in a quiescent state (G0 state in cell-cycle). The outer shell has the cells that are in an active state and can go therefore the natural cell life cycle (G1 S G2 M).

Comparison of previous models

The use of bottom-up and/or top-down approaches in the modeling of cellular systems.

Each approach as it's advantages and it's disadvantages. A top-down approach will allow the scientist with a valuable tool for generically describing the dynamics of a system, without going into much detail on how the particular components work. Those simulations mimic the overall behavior of the system, even if the rules underlying the behavior aren't mapped in the real system, although many times they have some sort of inspiration from the micro link. The top-down approach gains in generality but what it gains it looses in assertiveness as the models will be difficult to validate against real microlevel data, and will not have a defined field of application. On the other hand, this approach is particularly interesting as a tool to develop rapid conceptual frameworks of the problem in analysis and might macroscopically explain phenomenons for what the bottom-up approach still hasn't the power to explain.

The bottom-up approach allows scientists to model the complex diseases such as cancer in a level and resolution that can predict the correct “how’s” and “why’s” of drug action on the tumors. This can't be done without a comprehensive knowledge at the molecular level. This bottom-up approach has it's trade off in the form of more computational power needed to run the simulations, better understanding of the underlying molecular interactions and components, and a bigger variety of actors present in the simulation. These kinds of approach are time consuming and resource demanding, due to the high number of freedom degrees that are present in the study system.

These two different approaches reveal two pathways that are usable in the research of biological systems, and as the computational power needed is being made available by the industry, both pathways will end up converging somewhere in the middle.

From the bibliographic research that we’ve conducted we can now resume the main characteristics of what we’ve observed in the following table:

Table 1 – Mais aspects of the several approaches studied.

	Main Characteristics
Wishart et al. 2004	• Dynamic 2D cellular automaton • Not Cancer or Cellular Growth • Square Lattice • Drag & Drop Interface • Only intracellular dynamics • SBML databases • Doesn’t allow the prediction of properties • Java
Abott et al. 2006	• 3D cellular automaton • Cube lattice • Implements the hallmarks of cancer

	Main Characteristics
	• Extracellular dynamics • Simplified Intracellular dynamics • C++ / free license
Patel et al. 2001	• Hybrid 2D Cellular Automaton • Influence of Glucose, H^+ and Vascularization on Cancer growth • Differential equations used for Glucose and H^+ diffusion • C / DEC Alpha Unix Workstation
Kansal et al. 2000	• 3D Cellular Automaton • Voronoi Lattice • Brain Cancer (Specific) • IBM SP2 Parallel Computer (AIX) • 1.5 Million Lattices (minimum) • Large initial cell population

From the models observed, its clear that mainly the development of models tend to use some form of cellular automaton. They aren't pure CAs as they don't limit each cell to a vector state where all combinations are known and a set of rules very well established that are the same for each and every cell. Usually they implement some sort of stochastic factor and also they use some sort of continuum mathematical description of the environment where cells live, mainly in the form of ordinary differential equations.

From this table we also noticed that models tend to require high computational power. Exception made to the model presented by Wishart et al. although this is the only model that focuses entirely in the intracellular aspects of cellular dynamics although it doesn't model cell colony growth.

Different lattices are used in this modeled but it is noticed that researchers tend to prefer 3D environments as they mimic reality better.

From the analysis of this works and taking into account the limitations of computational power, time and resources that we had for this study we opted to try to bring into a simulation the ideas of the Abott et al. model that implement a subset of the hallmarks of cancer and add to this model the ideas of acid production that were described by Patel et al. The ideas of this project complement something that lacks on the former, as it doesn't implements the effect of the tumor cells in the environmental conditions. The production of acid acts as an inhibitor for cell growth, both normal and tumorous and therefore should be considered. The 3D modeling was discarded at this time as this extra layer of complexity in our model wouldn't allow us to develop a fully functional model, or even analyze it properly. The aim of our model was then to attempt to make a descriptive representation of the cancer growth phenomenon and evaluate the viability of using cellular automaton to mimic the real behavior of cancer growth.

3 CellCom: Model Proposal

It is our intention to produce a model that can describe the mechanism of tumor growth through the implementation of what is considered consensual in the formation of cancer cells and it's growth. From the previous readings we've outlined a description of the model that would incorporate the hallmarks of cancer with the evidence that cancer cells produce acid that will change the surroundings and affect the dynamics of normal and tumorous cells. We also included some aspects that interest us in respect to communication detection and communication patterns. In biological systems communication is processed by chemical signals and can broadly be characterized in four groups: Long distance communication includes point to point communication as for example neuronal signals are sent through to any point of the organism from the brain, and single to multi or multi to multi communication in the form of hormones that are segregated by glandules and then propagated in the body by the vascular system. Short range communication is also divided in two different aspects of communication, as it can contact-dependent, meaning that cells will only signal other cells they have physical contact with, or it can be short-range diffusion signaling as chemical signals will only affect those in the vicinity of the cell. In our model we want to model some of this aspects mainly we will include contact-dependent communication in the way cells can perform mitosis, we will have short range diffusion communication in the terms that cancer cells produce acid that will subsequently diffuse locally affecting both normal cells and tumor cells. We also implement a pseudo-long distance in the form of nutrients being fed by the vascular system.

We programmed a model of cell communication where cells are located in a 2D lattice. There positions will be fixed and cells wont have movement. The signals were modeled not as independent agents, but as properties of each lattice place, as the scales between cells and signals are several orders of magnitude different.

From the previous readings we had the idea of implementing some sort of molecular communication mechanism within the scope of nanomachines presented by Hyama et al. but due to the different scales at which cells and molecules operate this idea had to be discarded and was pushed to further developments.

Next, we present the description of model in detail, according to the protocol proposed by Grimm et al. (2006) called ODD (Overview, Design concepts, and Details).

Overview, Design Concepts and Details

Overview

Purpose

The purpose of the model CellCom is to illustrate how tumorous cells are generated from the genetic mutations that occur in normal cells, and to show how these mutations are affected by chemical signals from it's environment. Also the model will verify that even in a 2D cellular automaton that implements a set of properties and methods from biology, it is possible to test the dynamics of tumor formation and the overall tumor mass growth according to the literature models.

State Variables and Scales

The model comprises four levels that affect its overall behavior: Nutrients space, Acid space, Vascular space and Cell space.

Cells are described by a pseudo genome of phenotypic manifestations. As we consider the hallmarks of cancer to be the result of gene mutations, each cell has a pseudo genome that has equi-probable "genes" that will activate each of those hallmarks. Also this genome includes one extra gene to take into account genetic instability and this manifestation is also modeled has if it was induced by a gene mutation. Each cell is initialized with a copy of its parent cell genome. At this time the metastasis gene will not be modeled, as the mechanisms that lead to it are still not well understood. It would also require a much vast space, which is several orders of magnitude greater than the tumorous formation here studied.

The vascular system will be responsible for the distribution of nutrients across the tissue. At the initialization process only one vascular cell will be created in the same place has a normal cell. Then through the angiogenesis process triggered by a mutation in cells, the vascular system will be called to multiply and grow into those cells that have mutated.

The nutrients space only has one variable that represents the concentration of nutrients in each specific location. This concentration is an overall measure of all the nutrients existent in this place. We can imagine this entity to mimic the glucose concentration at each place and we choose to implement a serum concentration that can't go under 2.5 mM to allow cells to avoid hypoglycemic effects (Patel et al., 2001)

The acid space is similar to the nutrient space as it represents the acidity of the medium. The model is initialized admitting a pH of 7.4 in all cells. As tumor cells start producing acid then a simple model of diffusion will transport acid from high concentration cells to less concentrated ones.

We had to define two constant rates, *kRate* and *hRate*. *kRate* takes into account the consumption of nutrients by cells and *hRate* the production of acid by tumorous cells due to their anaerobic metabolism. The *kRate* for tumorous cells is a ten fold of the *kRate* for the normal cells and therefore we didn't implement a separate constant.

Table 2 – Parameters of the model and their default values.

Parameter	**Description**	**Value**
ProbCompeteNeighbor	When a Cell has the Insensitivity to Growth-Signals Gene activated it's daughter will compete with a neighbor and have this probability of success.	0.4
ProbDetectionDamageCells	This is the probability of detection that a gene is mutated. This check is made on G2 phase of the cell life-cycle.	0.97
ProbNormalGeneMutation	All genes have this equal probability of undergoing mutation in S phase of the cell life-cycle.	0.01
ResidualApoptosisProbability	This is the probability that a cell will have to go under Apoptosis that represents other factors not accounted by the model.	0.1
TelomereSize:	This is the number of divisions a cell can undergo before dieing. Cancer cells that have the Limitless Replicative Potential gene activated will ignore this and live forever.	10
VascularNutrients	This is the concentration in mM that the vascular system is capable of distributing to the surroundings of each cell to which it is connected.	5
WorldXSize	Dimension of the lattice in X	150
WorldYSize	Dimension of the lattice in Y	150

Process Overview and Scheduling

The model evolves in a discrete step manner, with each step corresponding to a possible complete life-cycle of the cell. We say "possible" because as we've seen, it is possible for a cell to enter a non-mitotic state G0 where it isn't dividing. As cells life cycle can have very different times for their life-cycles we can't map the time step to a specific amount of time. Some cells might have a life-cycle of 12h but others will have longer life-cycles or shorter, depending on the functions they perform in the organism. In our case we assume that each step of the simulation holds the time of a complete life-cycle. In each step, the model performs the following tasks:

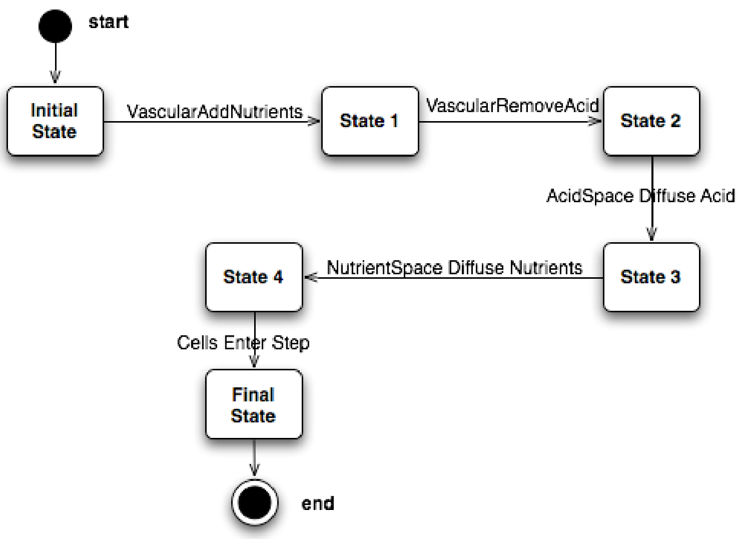

Figure 3 – Model State Diagram

The model at each step starts by ordering the vascular system to add nutrients to the tissue (state 1) and remove the acid produced by tumorous cells (state 2) then the model implements the diffusion models for acid and nutrients (State 3 and 4). At State 4 the preparation of the scenario where cells act is ready and the model orders each cell to perform the tasks in the step as we can see in the following figure.

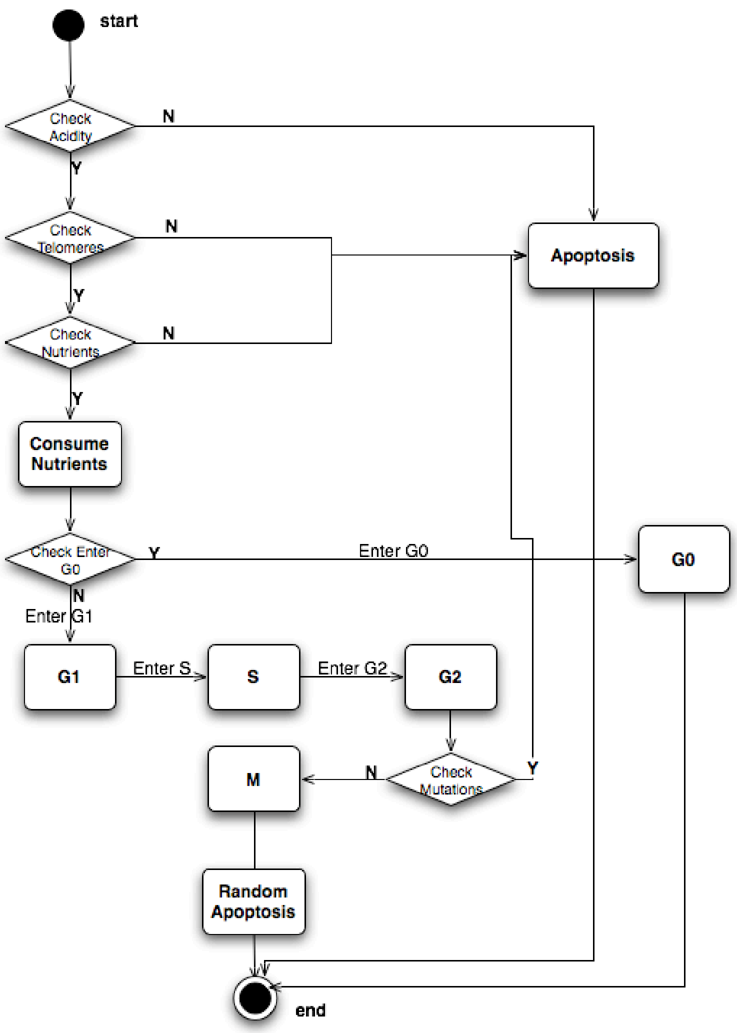

Figure 4 – Cell "Pseudo-Step" Diagram

When cells are ordered to perform their steps, they start by checking ("sensing") the environment for acidity and nutrients. Also they check to see if they can replicate by checking if it's telomere size is above 0. If the tests fail the cell enter apoptosis and will leave a blank space for other cells to grow. After that cells will remove a quantity of nutrients from the nutrient space and will check if they can enter the cell life-cycle. If they aren't allowed to enter the life-cycle, they will rest in a quiescent state G0 and then the step will end. If they can enter the normal life-cycle then they will go through the phases G1, S and G2. After this they will be checked for mutations in the genome and if mutations are detected then the cell will undergo apoptosis. Only if the cell escapes mutation detection will it then be allowed to go into the mitosis phase (M).

After this step all cells will eventually suffer apoptosis with a probability defined by the user.

Design Concepts

Genome

Although the hallmarks of cancer are the expression of gene mutations, they aren't mapped in reality to particular singular genes. Each might be the result of several mutations in real cancer tumors. As this genes aren't yet fully discovered and understood in our model we've decided to include a "genome" that represents each of the hallmarks of cancer in a manner that each "gene" represents one of the hallmarks, each one having an equal probability of mutating. Further more, the evidence of genetic instability isn't in itself a gene mutation, but a result of multiple mutations observed in real cancer masses. In this model the genetic instability is also modeled as a gene that will affect the probability of mutation of other cells by a factor of 10. This leads to a model genome that is no more than a vector of 6 integers which value will determine if a particular gene is mutated or not. As we won't model metastasis mutation this vector is composed of 5 hallmarks plus genetic instability.

Emergence

The tissue dynamics emerge from the behavior of the individual cells. Each cell has it's own life-cycle and rules on how each cell acts according to it's environment.

Adaptation

Cells can go into a quiescent state if their environment changes. If the pHs of the cell lattice falls bellow a certain value then the cell enters the G0 phase. This threshold is 7.1 for normal cells and 6.4 for cancerous cells. Cells that have the *induce angiogenesis* mutation also can send requests for vascular cells to proliferate in their direction if the nutrients in the lattice fall under a minimum value.

Sensing

Cells sense their environment in the form of chemical signals. In this model cells perceive nutrients concentration, pH and growth inhibitors.

Interaction

Cells act with each other by detecting the composition of the lattice they occupy. This can be nutrients or pH or the presence of other cells. Further, cancer cells can induce angiogenesis in the vascular system interacting at a longer distance than normal cells. Also, tumorous cells will stochastically compete with its neighbors for the occupancy of lattice locations.

Stochasticity

The model dynamics use several stochastic values to determine how cells will act. This is due to the fact that the CA isn't governed by a set of rules that respond in the same manner to a repetition of the same environment. Instead, the cell will probabilistic act in reaction to their surroundings. The features that require stochasticity values are defined by probabilities of occurrence and are defined and can be controlled by the user in the table of parameters.

Observation

The data is collected in tabular text files with the number of cancerous cells over time. Also graphs are produced that describe the percentage of cells with different mutations and also percentages of the number of mutations.

Details

Initialization

The model is initialized by the creation of a 2D NxN lattice and by the creation of the state vectors that will old information regarding nutrients concentration and acid concentration. At the beginning nutrients will be 0.0 at each location and the pH will be 7.4 for all locations.

Then one natural cell is created in the center of this lattice with a genome that has no mutations. At the same place one vascular cell is created and the initial quantity of nutrients is placed in this location. The vascular system cells can coexist with other cell types in the same lattice space, as they are considered independent.

Input

Nutrients Update

As the Cells in the model go through every step of their life, they have to consume resources. In our model resources are called nutrients and they are updated at each step of the simulation. The idea that we implemented in this model is that the concentration of nutrients at each vascular cell will always be constant. The idea is that the blood vessel will be able to transport nutrients to all microvessels equally without being perturbed by the size of the vascular system in the model. After the Nutrients Update they will be diffused by a simple diffusion mechanism discussed in the submodels section.

Acid Removal

Acid production is also removed by the vascular system. In our model the acid produced by cancer cells will diffuse by the same simple mechanism used by nutrients. At each step the vascular system will remove the excess acid in that location allowing the diffusion of high acid concentrations towards the vascular system to removal.

Submodels

Diffusion Model for Nutrients

In a steady state system, a Fick Law describes diffusion of a compound in a solution and the diffusion flux is related to a diffusion coefficient D that is characteristic of the species in question and proportional to the gradient of concentration that traverses the control volume. This relation as the following form for a one-dimensional control volume:

$$J_i = -D\frac{\delta C_i}{\delta x} \qquad (1)$$

This approach would lead us to have to perform a multitude of calculations including the integration of this equation on the entire discrete space demanding more power from the computational level and making the model to make some assumptions on frontier conditions. Therefore we tried a different approach to simulate the diffusion of nutrients. Once any perturbation in concentration would form a local gradient of concentration with it's neighbors, we'd assume that at each time step a locally steady-state would be achieved. This meant that at each step for each position in the lattice the step+1 concentration would be calculated as the cell concentration summed with the concentrations in the Moore neighbors cells and then averaged. This would make nutrients diffuse locally without the need to perform more complicate calculations.

Diffusion Model for H^+

The acid diffusion is implemented as in the nutrients space, with a small difference. The local steady state isn't achieved in every step but instead it is assumed that it would be achieved only after a finite number of steps (the model uses 5 time steps). This model means that the resulting increase or decrease of acid concentration will be only 20% of what it would if the diffusion was locally steady-state at each step.

Vascular Growth Model

The vascular system grows in response to cells that have the *sustained angiogenesis* mutation. In such cases if the mutated cell has a low concentration of nutrients surrounding it, it will send a signal to the nearest vascular cell calling them to replicate in it's direction. The vascular cell then calculates the direction toward the signal origin and one new vascular cell is created in it's Moore neighborhood only if that place is empty (of vascular cells) and it hasn't a vascular cell in that location Moore neighborhood other than it's parent. Although this allows, in some cases, the expansion of the vascular system from the trunk instead the leafs it creates structures that can somehow mimic the vascular system of real tissues.

Growth-Inhibitors Model.

We've implemented growth inhibitors by contact dependent signals. This means that a cell will enter G0 phase when all it's 8 surrounding lattice places are occupied. Normal cells won't go into their life-cycle if the count off cell in the Moore neighborhood is 8. Cells that have the *insensitivity to growth-inhibitory* mutation can escape growth inhibitors and will enter the G1 phase undergoing subsequent replication. Then the daughter cell will compete with one of the neighbors with a probability defined by the **ProbCompeteNeighbor** parameter.

Gompertzian Growth

The Gompertzian Growth is a population growth expression that is an exponential with a constant exponential. It assumes the form:

$$V = V_0 \times e^{\left(\frac{A}{B}\left(1-e^{-Bt}\right)\right)} \qquad (2)$$

Where *A* and *B* are equation parameters.

4 Experimentation and Results

In the experimentation of the model we've decided that we needed to explore the space of possibilities in a way that would make the model behave as closely as possible to the real understanding of how tumors grow. For that we've designed a set of experiments to search for local zones where tumor growth would seam similar to real tumor growth.

The proposed model was first run with a broad range of parameters to find local zones of interest in the space of solutions. After those local spaces have been identified the simulation was then run in a batch of tests in that local zones to verify that the outcome of the results where indeed due to the local zone parameters, and not from some stochastic behavior in one particular run.

As the model has some stochastic behavior in it's dynamics, sometimes the initial cells would mutate and would be detected rapidly. This would make those few cells enter apoptosis and the simulation would stop early. We've defined that for a run to be considered successful it would have to produce a tissue (tumorous or not) that would grow to the lattice boundaries or that would run for 2000 steps. Runs that didn't comply with this criteria where discarded.

The next figure shows the result of one of those runs.

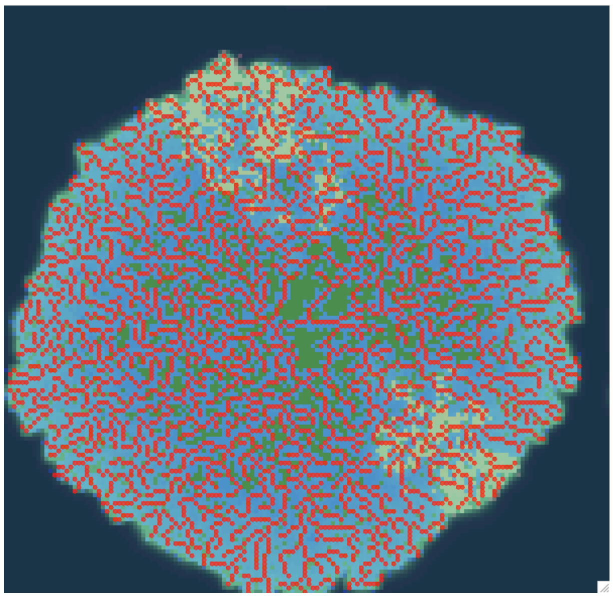

Figure 5 – Example of one model run.

In this figure we can see the all system modeled at the end of a run. The figure is composed of 4 layers that superimpose the cell space, the acid space, the nutrients space and the vascular system. For a better understanding of the results, next we present this four layer in a separate way.

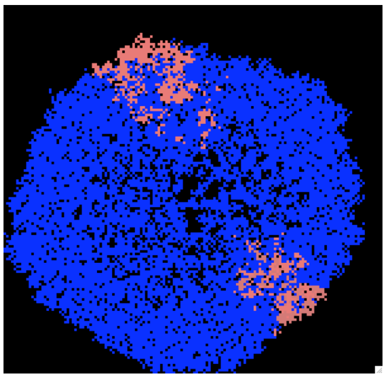

Figure 6 – Cell Space Layer

This figure represents the fully developed cancer cells in Blue. Pink cells are cells that had some kind of mutation but didn’t present all hallmarks of cancer. It is also visible that in the center region we can observe empty spaces because of acid concentration in this zones being higher then the threshold to sustain the existence of cells.

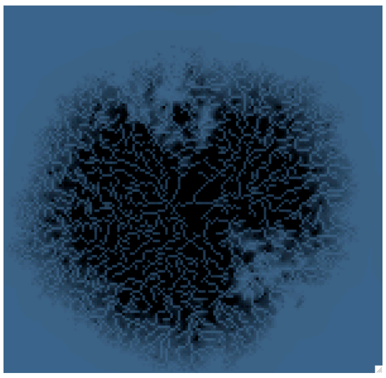

Figure 7 – Acidity Space

In this figure dark zones are more acid than blue zones. As the cancer growths inner zones tend to be more acid and that lead to the observed necrosis in figure 6. We can also observe that the vascular system, responsible for removing acid, is detected by the formed pattern because those zones have a higher pH (lower acidity).

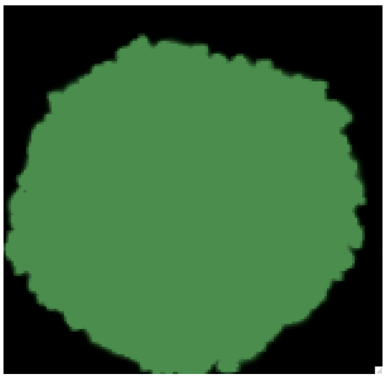

Figure 8 – Nutrients Space

As the vascular system develops due to the *sustained angiogenesis* mutation nutrients diffuse to all the tissue. This figures shows that in this set of parameters the cells are fed with sufficient nutrients.

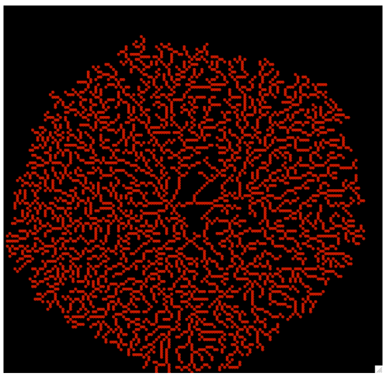

Figure 9 – Vascular Space

The vascular space shows that although it doesn't grow with a mapping to real vascular systems, as a new branch can emerge from any point at the tree and not just the leafs, it still produces natural-looking fractal-like structures that emerge endogenously. The pattern observed in this layer is also observed in the acidity layer as the vascular space is also responsible for removing acid from the tissue.

From the initial analysis we've defined a set of parameters that were then run in a batch process. These were the parameters described in table 2. The model was run 53 times and from this 6 where discarded and 47 used for further analysis. From this 47 runs we obtained the following statistics:

Table 3 – Statistics for the 47 runs.

	Average	StdDev
Number of Cancer Cells	5869	2833
Ticks	732	342

The representation of the runs in a graph indicates that the appearance of tumorous cells is sparse and probably is due to stochasticity of the model.

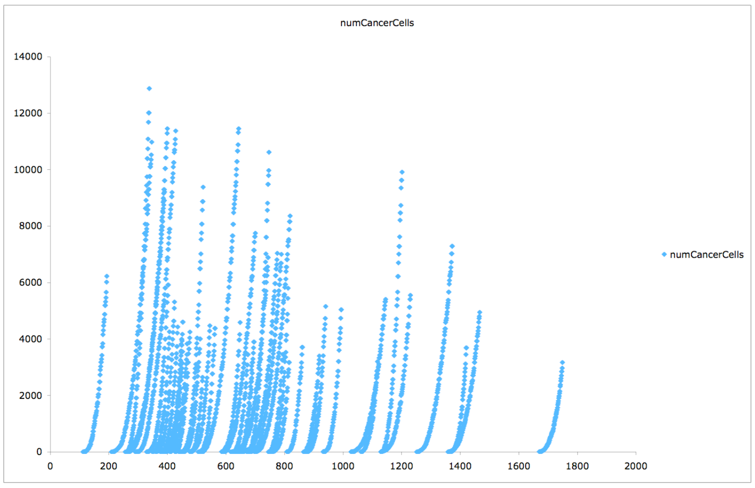

Figure 10 – Number of cancer cells on all 47 runs.

From the previous figure we can observe that although the emergence of tumorous cells is randomly sparse, the cancer growth of each run follows a similar growth. This fact leads us to discard the ticks where no cancer was observed in order to analyze the dynamics of the growth in a comparable manner. After representation of the 47 runs in a log-log graph we've ended obtaining the following representation:

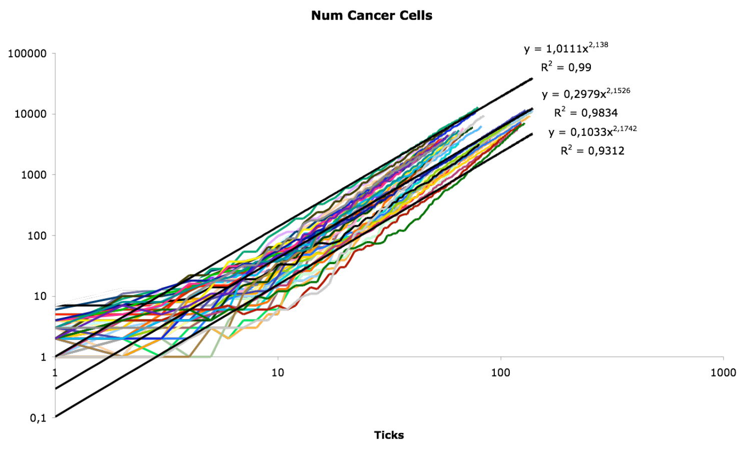

Figure 11 – log-log graph of the 47 runs.

The slope of the curves in this log-log figure is an indicator that the runs behave in a power law of the form:

$$y = a \times x^b \tag{3}$$

From the previous figure we can see that the cancer growth is ruled in a two different process. While the number of cancer cells is small, (under 10) the growth is approximately linear, but then the growth assumes a power law growth. The *b* parameter determines the slope of the log-log graph and as been calculated for the 47 parameters. We've calculated a value of **2,19** with a standard deviation of **0,21** for the exponent *b*. The *a* parameter is a factor of scale that we've found to be **0,42** with a standard deviation of **0,32.** This leads to the following equation with a confidence interval of 95%:

$$y = (0.42 \pm 0.64) \times x^{2.19 \pm 0.42} \tag{4}$$

The error in the *a* parameter means that there's a great dispersion of values. On the other hand the power parameter as a smaller error meaning that the behavior of the growth is very similar for the different runs.

These values show an interesting aspect of the dynamics of tumor growth in this model. The growth doesn't behave in a Gompertzian way as it was verified by some authors and it also doesn't grow linear as in the study of Brú et al., falling somewhere in between these two boundaries. This indicates that the inclusion of acidity in the model decreases the rate of growth but by itself might not be sufficient to explain the observed growths in *in vivo* and *in vitro* situations. It is also possible that the set of parameters used conditioned the results and other local zones of interest in the space of solutions, might need to be explored.

5 Discussion

We've modeled the growth of a tumorous tissue *in silico* using a hybrid cellular automaton. This differs from the traditional cellular automata by the inclusion of stochasticity ruled by probabilities that the user can define. The traditional cellular automata models use a very well defined set of rules and states that each agent can be in, usually in a reactive manner to the environment. In our case our agents are reactive to the environment but the decision process is stochastic in some aspects. Also in traditional CA all lattice places have those set of rules and states. In our approach the lattice is empty and agents are placed according to the dynamics of the model and only then they are part of the CA. Also if conditions for apoptosis are present, agents can be removed from the lattice.

This work showed that it is possible to achieve some descriptive understanding of the cancer growth, although the results on the dynamics fall between the models of Gompertzian growth and linear growth. We've found that in this case a power law describes the growth more appropriately. This implies that further research must be done in the mechanisms of the model and we've must then analyze some aspects that we think are pertinent. First, we need to do a scale analysis to ensure that the power law is maintained through a bigger environment. Secondly, we must investigate other zones of interest in the space of parameters. Thirdly, we must reflect if the 2D constrains in space are responsible for the observed dynamics as real tumors grow in 3D.

Also the implementation of a CA model as the base for the simulation has it's limitations, as it implies that discrete values occupy each lattice position. This discontinuity could be eliminated if some aspects of the CA would be modeled by differential equations with certain boundary conditions. This would increase the computational requirements for this model. Also the use of a 2D lattice might be responsible for the divergence between the power law observed and the other two models usually described in literature. Further work should include this aspect into account.

Our approach showed that with simple mechanisms to describe basic phenomena, the simulation could mimic, at least in a qualitatively manner, the real process of cancer growth.

We believe that the approach we took has a descriptive value and shows that the techniques used can be employed in practical cases of interest for the scientific community.

6 Further Research

This study presented a general overview how methodologies used in the field of complexity sciences can be applied to biology and particularly how cellular automata can be extended to show some insights on how biological systems evolve. Although this work showed that is possible to model cancer growth by these methodologies, further research should be done in ways that could improve the acceptance of these methodologies in this field and to achieve a greater quantitative understanding of the dynamics of tumor growth.

This model assumes that the cells live in a 2D tissue. This is obviously a limitation that the reality doesn't have. Further research should include a 3D lattice to investigate this problem, as the growth rate observed might be a result of a 2D geometry.

Research in different topologies of the lattice should be considered. Cells aren't squares in reality and other topologies should be considered. Voronoi spaces could be implemented as a solution. This Voronoi cells could me modeled in a 2D space or in a 3D.

This model doesn't explain the inner cell mechanisms as it pretends to replicate macroscopic phenotypic events. This model assumes for example, that induced angiogenesis is controlled by a single gene of our hypothetical genome. In reality angiogenesis is regulated by a variety of signals. Therefore future research should include some micro models of the pathways that produce these signals. Also chemical signaling in this model is considered instantaneous, which isn't verified in reality. Some sort of transport mechanism should be considered, probably with the application of concepts developed in the nanomachines fields.

Another aspect that should be considered in future studies is that in our model all phenotype switches are equal probable. That isn't realistic as some phenotypic manifestations are a cause of more gene mutations than others. Also the mapping of a phenotype manifestation to a single gene isn't exact and further research should be done to allow the inclusion of other genes in the model genome, and also to investigate the part of dumb DNA in the process.

Although our genes are equi-probable making all mutation sequences possible in the formation of cancer cells, further research should include mechanisms to detect which mutation sequences are more prevalent. Also the mechanisms at the several levels of abstraction that would be implemented should be chosen to mimic reality to the possible extent allowing a more approximate mapping between the model and reality.

The modeling of the vascular system in our model is very simple, allowing the growth of the capillaries from the nearest segment. The branches can therefore be formed from any segment of the vascular system and not just from the leaves, which is a departure from reality. In futures implementations this should be addressed allowing representing the vascular system as a network. This should allow further research on the way the angiogenesis is controlled by tumor cells.

7 References

Abott R.G.; Forrest, S.; Pienta, J. K.; Simulating the Hallmarks of Cancer; Artificial Life, Vol 12 (2006) , n.º4, 617 - 634

Alberts, B; Bray, D.; Johnson, A.; Lewis J.; Raff M.; Roberts, K.; Walter, P.: Essential Cell Biology, (1998) Graland Publishing inc. New York & London.

Brú, A.; Alberto, S.; Subiza, J.L.; García-Asejo, J.L.,; Brú, I.: The Universal Dynamics of Tumor Growth; Biophysical Journal, Volume 85 (Nov 2003) 2948-2961

Grimm, V. et al. ; A Standard Protocol for Describing Individual-based and Agent-based Models: Ecological Modelling 198 (2006) 115-126

Hanahan, D.; Weinberg, R. A.: The hallmarks of cancer. Cell, 100 (2000), 57 – 70.

Harel, D.; Comprehensive and Realistic Modeling of Biological Systems; Proceedings of the 2006 Winter Simulation Conference.

Hiyama, S.; Moritani, Y.; Suda, T.; Egashira, R.; Enomoto, A.; Moore, M.; Nakano, T.: Moelcular Communication; Proc. of the 2005 NSTI Nanotechnology Conference, 2005

Kansal, A.R; Torquato, S.; Harsh IV, G.R.; Chiocca, E.A.; Deisboeck, T.S; Simulated Brain Tumor Growth Dynamics Using a Three-Dimensional Cellular Automaton; J. theor. Biol. (2003) 203, 367-382

Parkin, D. M.; Bray, F., Ferlay, J.; Pisani, P. Global cancer statistics, 2002. CA: A Cancer Journal for Clinicians, 55, (2005) 74 – 108.

Patel, A.A.; Gawlinksi, E.T.; Lemieux, S.K.; Gatenby, R.A.: A Cellular Automaton Model of Early Tumor Growth and Invasion: The Effects of Native Tissue Vascularity and Increased Anaerobic Tumor Metabolism; J. theor. Biol. (2001) 213, 315-331

Wishart, D.; Yang, R.; Arndt D.; Tang, P.; Cruz, J.: Dynamic cellular automata: an alternative approach to cellular simulation; In Silico Biology 4, 0015 (2004); ©2004, Bioinformation Systems e.V.

Whitesides, G.M., The Once and Future Nanomachine; Scientific American, Sep, 2001.

Annexes

```java
/*
 * Model.java
 *
 * Created on May 18, 2007, 5:38 PM
 *
 */

package cellcom;
import java.awt.Color;
import java.util.ArrayList;
import java.util.Vector;
import uchicago.src.sim.analysis.DataRecorder;
import uchicago.src.sim.analysis.OpenSequenceGraph;
import uchicago.src.sim.analysis.Sequence;
import uchicago.src.sim.engine.BasicAction;
import uchicago.src.sim.engine.Schedule;
import uchicago.src.sim.engine.SimInit;
import uchicago.src.sim.engine.SimModelImpl;
import uchicago.src.sim.gui.DisplaySurface;
import uchicago.src.sim.gui.Object2DDisplay;
import uchicago.src.sim.space.Object2DGrid;
import uchicago.src.sim.space.Object2DTorus;
/**
 * Model Class for CellCom. Implements the SimModelImpl rquired by Repast to run
 * the simulation.
 * @author david
 */
public class Model extends SimModelImpl {
    // Repast Parameters
    private DataRecorder recorder;
    private int worldXSize=150;
    private int worldYSize=150;
    private Schedule schedule;

    /**
     * Space that holds the simulation Cells
     */
    public Object2DTorus cellSpace;
    /**
     * Space that holds the vascular system
     */
    public Object2DTorus vascularSpace;
    /**
     * Space that holds the nutrients
     */
    public Object2DTorus nutrienteSpace;
    /**
     * Space that is used to update the nutrients space with the simple diffusion
mechanism
     */
    public Object2DTorus nutrienteNextSpace;
    /**
     * Space used to draw the nutrients space
     */
    public Object2DTorus moleculeSpaceSpace;
    /**
     * Space that holds de Acid Space
     */
    public Object2DTorus acidSpace;

    private DisplaySurface dsurf;
    private OpenSequenceGraph numCellPhenotype;
    private OpenSequenceGraph numCellMutations;
    /**
     * Counter for the number of cells produced in the model
     */
    public int cellNumber=0;
    /**
     * Counter for the number of vascular cells produced in the simulation
     */
    public int vascularNumber=0;

    ArrayList cells;
    ArrayList vascular;
    ArrayList nutrients;
    ArrayList acids;


    // Parameters of the stocastic Model...
    private double VascularNutrients=5; // mM
    /**
     * Probability of detection of damaged cells
     */
    public double probDetectionDamageCells=0.97;
    /**
     * Probability of residual apoptosis
     */
    public double residualApoptosisProbability=0.1;
    /**
     * Probability of comepetition with neighbor cells
     */
    public double probCompeteNeighboor=0.4;
    /**
     * Probability of individual gene mutation
     */
    public double probNormalGeneMutation=0.01;
    /**
     * Multipling factor for genetic instability factor
     */
    public double multIncreaseGeneMutation=10;

    /**
     * Telomere Size - limits the number of mitosis a cell can undergo
     */
    public int telomereSize=10;

    /**
     * Creates a new instance of Model
     */
    public Model() {
    }

    /**
     * Passes the model parameters to the Repast Controller
     * @return Array of Parameters
     */
    public String[] getInitParam() {
        return new String[]{
            "WorldXSize",
            "WorldYSize",
            "ProbDetectionDamageCells",
            "ResidualApoptosisProbability",
            "ProbCompeteNeighboor",
            "ProbNormalGeneMutation",
            "TelomereSize",
            "VascularNutrients"
        };
    }
//---------------------------------------------------------------------------------------------- BEGIN
    /**
     * Begin method - Called by Repast controller
     */
    public void begin() {
        System.out.println("begin()");
        buildModel();
        buildSchedule();
        buildDisplay();

        dsurf.display();
        numCellPhenotype.display();
        numCellMutations.display();
    }

    // -------------------------------------------------------------------------- BUILD DISPLAY
    private void buildDisplay() {
        System.out.println("buildDisplay()");

        Object2DDisplay cellDisplay = new Object2DDisplay(cellSpace);
        cellDisplay.setObjectList(cells);

        Object2DDisplay vascularDisplay = new Object2DDisplay(vascularSpace);
        vascularDisplay.setObjectList(vascular);

        Object2DDisplay nutrientsDisplay = new Object2DDisplay(moleculeSpaceSpace);
        nutrientsDisplay.setObjectList(nutrients);

        Object2DDisplay acidDisplay = new Object2DDisplay(acidSpace);
        acidDisplay.setObjectList(acids);


        dsurf.addDisplayableProbeable(cellDisplay,"Cells");
        dsurf.addDisplayableProbeable(acidDisplay,"pH");
        dsurf.addDisplayableProbeable(nutrientsDisplay,"Nutrients");
        dsurf.addDisplayableProbeable(vascularDisplay,"Vascular");
        dsurf.setBackground(Color.BLACK);
        /*
         *  ------------------------------------------------------------------
PERCENTAGE OF CELLS WIHT EACH TYPE OF MUTATION
         */
        numCellPhenotype.addSequence("Genetic Intability", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.geneticInstability){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );

        numCellPhenotype.addSequence("Evade Apoptosis", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.evadeApoptosis){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );

        numCellPhenotype.addSequence("Sustained Agiogenesis", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.sustaindedAgiogenisi){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );

        numCellPhenotype.addSequence("Ignore Growth Inhibit", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.ignoreGrowthInhibit){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
```

```java
            }
        }
        );

        numCellPhenotype.addSequence("Limitless Replication", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.limitlessReplication){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );

        numCellPhenotype.addSequence("Growth Signal", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.growthSignal){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );

        /*
         *  ----------------------------------------------------------------------
PERCENTAGE OF CELLS WITH X NUMBER OF MUTATIONS
         */
        numCellMutations.addSequence("0 Mutations", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.getGenome()==0){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );
        numCellMutations.addSequence("1 Mutations", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.getGenome()==1){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );
        numCellMutations.addSequence("2 Mutations", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.getGenome()==2){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );
        numCellMutations.addSequence("3 Mutations", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.getGenome()==3){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );
        numCellMutations.addSequence("4 Mutations", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.getGenome()==4){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );
        numCellMutations.addSequence("5 Mutations", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.getGenome()==5){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );
        numCellMutations.addSequence("6 Mutations", new Sequence(){
            public double getSValue(){
                double num=0.0;
                for (int i = 0; i < cells.size(); i++) {
                    Cell a = (Cell)cells.get(i);
                    if (a.getGenome()==6){
                        num++;
                    }
                }
                num=100*num/cells.size();
                return num;
            }
        }
        );

    }
    //
-------------------------------------------------------------------------------------------
-- BUILD MODEL
    private void buildModel() {

        System.out.println("buildModel()");
        cellSpace = new Object2DTorus(this.getWorldXSize(),this.getWorldYSize());
        vascularSpace = new Object2DTorus(this.getWorldXSize(), this.getWorldYSize());
        nutrienteSpace = new Object2DTorus(this.getWorldXSize(), this.getWorldYSize());
        nutrienteNextSpace = new Object2DTorus(this.getWorldXSize(), this.
getWorldYSize());
        moleculeSpaceSpace = new Object2DTorus(this.getWorldXSize(), this.
getWorldYSize());
        acidSpace = new Object2DTorus(this.getWorldXSize(),this.getWorldYSize());

        int nx = (int)(this.getWorldXSize()/2);
        int ny = (int)(this.getWorldYSize()/2);

        // Inicaliza espaço de Nutrientes

        for (int i = 0; i < nutrienteSpace.getSizeX(); i++) {
            for (int j = 0; j < nutrienteSpace.getSizeY(); j++) {
                nutrienteSpace.putValueAt(i,j,0.0);
                nutrienteNextSpace.putValueAt(i,j,0.0);

                MoleculeSpace a1 = new MoleculeSpace(this);
                a1.setXY(i,j);
                moleculeSpaceSpace.putObjectAt(i,j,a1);
                nutrients.add(a1);

                Acid a2 = new Acid(this);
                a2.setXY(i,j);
                a2.setPH(7.4);
                acidSpace.putObjectAt(i,j,a2);
                acids.add(a2);
            }
        }

        // Add 1st Cell
        Cell c1 = new Cell(this);
        c1.setXY(nx,ny);
        cellSpace.putObjectAt(nx,ny,c1);
        cells.add(c1);

        // Add 1st Vascular
        Vascular v1 = new Vascular(this);
        v1.setXY(nx,ny);
        v1.diffuseNutrients(this.getVascularNutrients()*15);
        vascularSpace.putObjectAt(nx,ny,v1);
        vascular.add(v1);
        // Add 1st Nutrient Concentration...

//          nutrienteSpace.putValueAt(nx,ny,this.getVascularNutrients()*15);


        //------------------------------------------------------------------------------------------RECORDER

        recorder = new DataRecorder("./data.txt",this);
        recorder.createNumericDataSource("numCancerCells", this, "getNumCancerCells");


    }

//------------------------------------------------------------------------------------------------------- SCHEDULE

    private void buildSchedule() {
        schedule = new Schedule();
        schedule.scheduleActionBeginning(0, this, "mainAction");
//              schedule.scheduleActionBeginning(0,this,);

        schedule.scheduleActionAtInterval(1, new BasicAction() {
            public void execute() {
                recorder.record();
            }
        });

        schedule.scheduleActionAtInterval(10,this,"updatePlots");

        schedule.scheduleActionAtEnd(recorder , "writeToFile");
    }

    /**
     * Step Action that the model has to run each tick
     */
    public void mainAction(){
        // Vascular... actualizar valores de nutrientes no sistema vascular

//        long time1=System.nanoTime();

        if (stopSimulation()){
            this.stop();
        }

        // Difuse Nutrientes;
        for (int i = 0; i < vascular.size(); i++) {
            Vascular v = (Vascular)vascular.get(i);
            double vh=nutrienteSpace.getValueAt(v.getX(),v.getY());
            v.diffuseNutrients(this.getVascularNutrients()+vh);
            Acid a = (Acid)acidSpace.getObjectAt(v.getX(),v.getY());
            a.setPH(7.4);
        }

        this.diffuseNutrients();
        this.diffuseAcid();


        /* Celulas Entrar no ciclo de vida
         *  verificam nível de nutrientes
         *      Se < a => pedem Vascular
         *      Se a > e < b estão em quiscent state
         *      Se > b mitose
         */
        int n=cells.size();
        uchicago.src.sim.util.SimUtilities.shuffle(cells);
        for (int i = n-1; i >= 0; i--) {
            Cell a = (Cell)cells.get(i);

            a.step();
        }







//        System.out.println(System.nanoTime()-time1);


    }
    /**
     * Updates the Plots of the simulation
     */
    public void updatePlots(){

        dsurf.updateDisplay();
        numCellPhenotype.step();
        numCellMutations.step();

    }
    /**
     * Diffusion sub-model that performs before each individual cell step
     */
    public void diffuseNutrients(){
        // Difundir nutrientes
        for (int i = 0; i < this.getWorldXSize(); i++) {
            for (int j = 0; j < this.getWorldYSize(); j++) {
                Vector viz=nutrienteSpace.getMooreNeighbors(i,j, false);
                double soma=0.0;
                for (int k = 0; k < viz.size(); k++) {
                    soma+=((Double)viz.get(k)).doubleValue();
                }
                soma+=nutrienteSpace.getValueAt(i,j);
                soma = soma / 9.0;
                nutrienteNextSpace.putValueAt(i,j,soma);
            }
        }
        for (int i = 0; i < nutrienteSpace.getSizeX(); i++) {
            for (int j = 0; j < nutrienteSpace.getSizeY(); j++) {
                nutrienteSpace.putValueAt(i,j,nutrienteNextSpace.getValueAt(i,j));
            }
        }
    }
    /**
     * Diffusion of Acid sub-model. Prepares Acid space to be interpreted by cells in
     * their individual steps
     */
    public void diffuseAcid(){
        for (int i = 0; i < this.getWorldXSize(); i++) {
            for (int j = 0; j < this.getWorldYSize(); j++) {
                Vector viz=acidSpace.getMooreNeighbors(i,j, false);
                double soma=0.0;
                for (int k = 0; k < viz.size(); k++) {
                    Acid a = (Acid)viz.get(k);
                    soma+=a.getConcH();
                }
                Acid eu = (Acid)acidSpace.getObjectAt(i,j);

                soma+=eu.getConcH();
                soma = soma / 9.0;

                eu.setNextConcH(eu.getConcH()+0.2*(soma-eu.getConcH()));
            }
        }
        for (int i = 0; i < acidSpace.getSizeX(); i++) {
            for (int j = 0; j < acidSpace.getSizeY(); j++) {
                Acid eu = (Acid)acidSpace.getObjectAt(i,j);
                eu.setConcH(eu.getNextConcH());
            }
        }
    }
    /**
     * Getter for the number of cancer cell
     * @return number of cancer cells
     */
    public int getNumCancerCells(){
        int out=0;
        for (int i = 0; i < cells.size(); i++) {
            Cell a = (Cell)cells.get(i);
            if (a.getGenome()==6){
                out++;
            }
        }
        return out;
    }

    /**
     * Stops the simulation if certain conditions are met
     * @return Returns true if conditions are met.
     */
    public boolean stopSimulation(){

        if (cells.size()==0){
            return true;
        }

        for (int i = 0; i < this.getWorldXSize(); i++) {
            if (cellSpace.getObjectAt(i,0)!=null){
                return true;

            }
            if (cellSpace.getObjectAt(i,this.getWorldYSize()-1)!=null){
                return true;
            }
        }
        for (int i = 0; i < this.getWorldYSize(); i++) {
            if (cellSpace.getObjectAt(0,i)!=null){
                return true;
            }
            if (cellSpace.getObjectAt(this.getWorldXSize()-1,i)!=null){
                return true;
            }
        }
        if (this.getTickCount()>2000){
            return true;
        }

        return false;
```

```java
    }

//
----------------------------------------------------------------------------------------------
----- SETUP
    /**
     * Setup Method - Called by the repast controller
     */
    public void setup() {

        cellSpace=null;
        vascularSpace=null;

        nutrienteSpace=null;
        nutrienteNextSpace=null;

        moleculeSpaceSpace=null;
        acidSpace=null;

        cells=null;
        vascular=null;
        nutrients=null;
        acids=null;
        schedule=null;

        if (dsurf!=null){
            dsurf.dispose();
        }
        dsurf=null;

        if (numCellPhenotype!=null){
            numCellPhenotype.dispose();
        }
        numCellPhenotype=null;

        if(numCellMutations!=null){
            numCellMutations.dispose();
        }
        numCellMutations=null;

        System.gc();

        cells = new ArrayList();
        vascular= new ArrayList();
        nutrients = new ArrayList();
        acids = new ArrayList();


        schedule = new Schedule();
        dsurf = new DisplaySurface(this, "CellCom");
        numCellPhenotype = new OpenSequenceGraph("% Cell With Phenotype", this);
        numCellMutations = new OpenSequenceGraph("% Of Cell Wiht X Mutations", this);

        registerDisplaySurface("CellCom",dsurf);
        this.registerMediaProducer("Plot", numCellPhenotype);
        this.registerMediaProducer("Plot2", numCellMutations);

//        System.out.println("setup()");
    }

    /**
     * Getter for the step scheduler. Called by repast controller
     * @return Returns a Schedule
     */
    public Schedule getSchedule() {
        return schedule;
    }

    /**
     * Getter for the name of the simulation
     * @return String with the name of the simulation
     */
    public String getName() {
        return "CellCom";
    }

    // ---------------------------------------------------------------------------------------SETTERS AND GETTERS


    /**
     * Getter for the ProbCompeteNeighboor parameter
     * @return Probability
     */
    public double getProbCompeteNeighboor() {
        return probCompeteNeighboor;
    }

    /**
     * Getter for the ProbDetectionDamageCells parameter
     * @return Probability
     */
    public double getProbDetectionDamageCells() {
        return probDetectionDamageCells;
    }

    /**
     * Getter for the TelomereSize parameter
     * @return Telemore Size
     */
    public int getTelomereSize() {
        return telomereSize;
    }

    /**
     * Getter for the vascular system arraylist
     * @return ArrayList with Vascular Objects
     */
    public ArrayList getVascular() {
        return vascular;
    }

    /**
     * Getter for the XX world Size
     * @return XX world size
     */
    public int getWorldXSize() {
        return worldXSize;
    }

    /**
     * Getter for the YY world size
     * @return YY world size
     */
    public int getWorldYSize() {
        return worldYSize;
    }

    /**
     * Setter for the ProbCompeteNeighboor parameter
     * @param probCompeteNeighboor Probability
     */
    public void setProbCompeteNeighboor(double probCompeteNeighboor) {
        this.probCompeteNeighboor = probCompeteNeighboor;
    }

    /**
     * Setter for the ProbDetectionDamageCells parameter
     * @param probDetectionDamageCells Probability
     */
    public void setProbDetectionDamageCells(double probDetectionDamageCells) {
        this.probDetectionDamageCells = probDetectionDamageCells;
    }

    /**
     * Setter for the TelemoreSize parameter
     * @param telomereSize Telemore Size
     */
    public void setTelomereSize(int telomereSize) {
        this.telomereSize = telomereSize;
    }
```

```java

    /**
     * Setter for the WorldXSize parameter
     * @param worldXSize Word X size
     */
    public void setWorldXSize(int worldXSize) {
        this.worldXSize = worldXSize;
    }

    /**
     * Setter for the WorldYSize parameter
     * @param worldYSize World Y Size
     */
    public void setWorldYSize(int worldYSize) {
        this.worldYSize = worldYSize;
    }

    /**
     * Setter for the VascularNutrients Parameter
     * @param VascularNutrients Vascular Concentration parameter
     */
    public void setVascularNutrients(double VascularNutrients) {
        this.VascularNutrients = VascularNutrients;
    }

    /**
     * Getter for the VascularNutrients Parameter
     * @return Vascular Nutrients Parameter Concentration
     */
    public double getVascularNutrients() {
        return VascularNutrients;
    }

    /**
     * Getter for the ProbNormalGeneMutation Parameter
     * @return Probability
     */
    public double getProbNormalGeneMutation() {
        return probNormalGeneMutation;
    }

    /**
     * Setter for the ProbNormalGeneMutation Parameter
     * @param probNormalGeneMutation Probability
     */
    public void setProbNormalGeneMutation(double probNormalGeneMutation) {
        this.probNormalGeneMutation = probNormalGeneMutation;
    }

    /**
     * Setter for the ResidualApoptosisProbability parameter
     * @param residualApoptosisProbability Probability
     */
    public void setResidualApoptosisProbability(double residualApoptosisProbability) {
        this.residualApoptosisProbability = residualApoptosisProbability;
    }

    /**
     * Getter for the ResidualApoptosisProbability Parameter
     * @return Probability
     */
    public double getResidualApoptosisProbability() {
        return residualApoptosisProbability;
    }

    /**
     * Main method. Called when simulation is ran.
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
//        System.out.println("main()");
        SimInit init = new SimInit();
        Model model = new Model();
        init.loadModel(model, "", false);
    }
```

```




}

```

```java
/*
 * Cell.java
 *
 * Created on May 18, 2007, 4:46 PM
 *
 */

package cellcom;

import java.awt.Color;
import java.util.Vector;
import uchicago.src.sim.gui.Drawable;
import uchicago.src.sim.gui.SimGraphics;

/**
 * Cell Class.
 * @author David Rodrigues
 */
public class Cell implements Drawable {
    // Mutations that a Ceel can Undergo
    /**
     * Defines de state of the gene <I>Growth Signal</I>
     */
    public boolean growthSignal=false;
    /**
     * Defines the state of the gene <I>Sustained Angiogenisis</I>
     */
    public boolean sustaindedAgiogenisi=false;
    /**
     * Defines the state of the gene <I>Ignore Growth Inhibitor</I>
     */
    public boolean ignoreGrowthInhibit=false;
    /**
     * Defines the State <I>Evade Apoptosis</I>
     */
    public boolean evadeApoptosis=false;
    /**
     * Defines the State of the gene mutation <I>Limitless Replication</i>
     */
    public boolean limitlessReplication=false;

    /**
     * Defines the state of the mutation <I>Genetic Instability</i>
     */
    public boolean geneticInstability=false;
    /**
     * Vector of the genome of the
     */
    public int[] genome={0,0,0,0,0,0};
    /**
     * Vector of the Genome to use during the Cell life-cycle
     */
    public int[] genomeDaughter={0,0,0,0,0,0};
    private int x;
    private int y;
    private Color cor;
    private Model modelo;
    /**
     * Indicates the fase the cell is in
     */
    public int fase=2;
    private int faseG1hold;
    int telomeres;
    String who="Cell ";
    /**
     * The Value of pH at which the cells dies due to acidity
     */
    public double pHDeath=6.8;
    /**
     * Defines the pH below which the cell enters a quiscent state
     */
    public double pHQuisc=7.1;
    /**
     * Defines the lowest level the nutrients can have before making the cell die.
     */
    public double nutThreshold=2.5;
    /**
     * The rate of consumption of Nutrients
     */
    public double kRate=0.01;
    /**
     * The rate of production of Acid
     */
    public double hRate=0.1;

    /* Fases do ciclo de vida da Célula
     *  1 - G0 a célula não tenta dividir-se
     *  2 - G1
     *  3 - S - Sintese de novo DNA
     *  4 - G2 Ponto de verificação genética.
     *  5 - M
     */


    /**
     * Creates a new instance of Cell
     * @param modelo Reference to the main mode in which the cell is running.
     */
    public Cell(Model modelo) {
        this.modelo=modelo;
        this.telomeres=modelo.getTelomereSize();
        this.who+=modelo.cellNumber++;
    }

    /**
     * method to draw the cells in the repast display
     * @param g the SimGraphics that defines how the object will be drawn in the
display.
     */
    public void draw(SimGraphics g) {
        int soma=0;
        for (int i = 0; i < genome.length; i++) {
            soma+=genome[i];
        }

        if (soma < genome.length){
            this.cor=new Color(100+(155/genome.length)*soma,100,100);
        } else {
            this.cor=new Color(0,0,255);
        }


        if (this.cor==null){
            this.cor = Color.DARK_GRAY;
        }


        g.drawRect(this.cor);
    }

    /**
     * Get the X position of the cell in the Cell Space
     * @return Returns th XX coordinates
     */
    public int getX() {
        return this.x;
    }

    /**
     * Get the X position of the cell in the Cell Space
     * @return Returns the YY coordinates
     */
    public int getY() {
        return this.y;
    }
    /**
     * Sets the XX and YY coordinates of the Cell
     * @param x The X coord.
     * @param y The Y coord.
     */
    public void setXY(int x, int y){
        this.x=x;
        this.y=y;
    }

    /**
     * Sequence that executes the cell step at each tick in the simulation
     */
    public void step(){



        // Check Acidity
        if (!checkAcidity()){
            enterApoptosis();
            return;
        }


//          System.out.println("step ");
        if (this.telomeres<=0){
            enterApoptosis();
            return;
        }

        // Check Nutrients
        if (!checkNutrients()){
            return;
        }



        // Consume Nutrients and Produce Acid
        consumeNutrients();



        enterG0Fase();


        if (this.fase!=1){

            enterG1fase();

            enterSFase();

            if (enterG2fase()){

                enterMFase();
            }

        }


        // Residual Apoptosis Probability...
        if (Math.random()<modelo.getResidualApoptosisProbability()){
            enterApoptosis();
            return;
        }
    }

    /**
     * Checks if the acidity in the Cell position is abobe the minimum threshold
     * @return True if Acidity is above minimum
     */
    public boolean checkAcidity(){

        this.pHDeath=6.8;
        if (this.getGenome()==6){
            this.pHDeath=6.0;
        }


        Acid a=(Acid)modelo.acidSpace.getObjectAt(this.x,this.y);

        if (a.getPH()<this.pHDeath){
            return false;
        }

        return true;

    }

    /**
     * Checks the nutrient space to see if the value is above a minimum threshold
     * Induces Apoptosis if necessary and sets the fase variable.
     * @return Return True or False
     */
    public boolean checkNutrients(){
        boolean out=true;
        double nutHere=((Double)modelo.nutrienteSpace.getValueAt(x,y)).doubleValue();

        if (nutHere<this.nutThreshold){
            // XXXX Alterar este Valor de threshold Minimo...
            if (sustaindedAgiogenisi){
                induceAngiogenesis();
                out=true;
            } else  {
                enterApoptosis();
                this.fase=1;
                out=false;
            }
        }
        return out;
    }

    public void consumeNutrients(){


        // Nutrients
        double nutHere=((Double)modelo.nutrienteSpace.getValueAt(x,y)).doubleValue();

        this.kRate=0.01;
        if (this.isCancer(this)){
            this.kRate=10*this.kRate;
        }

        double nutConsumed=this.kRate;

//        System.out.println(nutHere);
        nutConsumed=nutHere*Math.exp(-1*nutConsumed);
//        System.out.println(nutConsumed);
        modelo.nutrienteSpace.putValueAt(x,y,nutConsumed);

        // Acid Production only if there's mutation

        if (getGenome()==6){
            Acid a = (Acid)modelo.acidSpace.getObjectAt(this.x,this.y);
            double acidHere=a.getConcH();
//        System.out.println(acidHere);

            acidHere=acidHere*Math.exp(this.hRate);
//        System.out.println(acidHere);
            a.setConcH(acidHere);
        }

    }

    /* ---------------------------------------------------------------------------------
Cell Life Cycle
     *  G0 | G1 S G2 M
     *
     */
    public void enterG0Fase(){
        this.pHQuisc=7.1;
        if (this.getGenome()==6){
            this.pHQuisc=6.4;
        }

        Acid a=(Acid)modelo.acidSpace.getObjectAt(this.x,this.y);

        if (a.getPH()<this.pHQuisc){
            this.fase=1;
        }

        if (ignoreGrowthInhibit){
            this.fase=0;
            return;
        }

        Vector viz = modelo.cellSpace.getMooreNeighbors(this.x,this.y,false);
        if (viz.size()==8){
            this.fase=1;
        } else {
            this.fase=0;
        }
    }
    public void enterG1fase(){
//        XXXX Fase de Espera da Céula... Provavelmente não é para implementar...
        this.fase=2;
    }
    public void enterSFase(){
        //Sintetizar o genoma de uma filha com probabilidade de mutação...
        this.fase=3;
        genomeDaughter=genome;
        for (int i = 0; i < genomeDaughter.length; i++) {
            if (genomeDaughter[i]==0){
                if (geneticInstability){
//                    System.out.println("Genetic Instability");
                    if (Math.random()< modelo.probNormalGeneMutation*modelo.multIncreaseGeneMutation){
                        genomeDaughter[i]=1;
                    }
                } else {
                    if (Math.random()< modelo.probNormalGeneMutation){
                        genomeDaughter[i]=1;
                    }
                }
            }
        }
        if (!limitlessReplication){
            this.telomeres--;
//            System.out.println(this.who+": telemerer shortenning");
        }
    }
    public boolean enterG2fase(){
        // XXX Verifica se encontra células cancerigenas e marca-as para Apoptose...
        this.fase=4;

        boolean killThis=false;
        if (!evadeApoptosis){
            for (int i = 0; i < genomeDaughter.length; i++) {
                if (Math.random()<modelo.getProbDetectionDamageCells()){
                    if (genomeDaughter[i]==1){
                        killThis=true;
                    }
                }
            }
        }
        if (killThis){
            this.enterApoptosis();
        }
        return !killThis;
    }
    public void enterMFase(){
        // Uma nova célula
        this.fase=5;
        Cell a = new Cell(modelo);
        a.genome=this.genomeDaughter.clone();
        a.setBoleanGenes();

        int nx=this.x+(int)(Math.random()*3)-1;
        int ny=this.y+(int)(Math.random()*3)-1;
        nx = (nx + modelo.getWorldXSize()) % modelo.getWorldXSize();
        ny = (ny + modelo.getWorldYSize()) % modelo.getWorldYSize();

        if (!ignoreGrowthInhibit){
            while(modelo.cellSpace.getObjectAt(nx,ny)!=null){
                nx=this.x+(int)(Math.random()*3)-1;
                ny=this.y+(int)(Math.random()*3)-1;
                nx = (nx + modelo.getWorldXSize()) % modelo.getWorldXSize();
                ny = (ny + modelo.getWorldYSize()) % modelo.getWorldYSize();
            }
            a.setXY(nx,ny);
            modelo.cellSpace.putObjectAt(nx,ny,a);
            modelo.cells.add(a);
            this.genome=this.genomeDaughter.clone();

        } else {
            //XXX Probability of Success..
            if (modelo.cellSpace.getObjectAt(nx,ny)!=null){
                if (Math.random()<modelo.getProbCompeteNeighboor()){
                    Cell b = (Cell)modelo.cellSpace.getObjectAt(nx,ny);
                    b.enterApoptosis();
                    a.setXY(nx,ny);
                    modelo.cellSpace.putObjectAt(nx,ny,a);
                    modelo.cells.add(a);
                    this.genome=this.genomeDaughter.clone();
                } else {

                }
            } else {
                a.setXY(nx,ny);
                modelo.cellSpace.putObjectAt(nx,ny,a);
                modelo.cells.add(a);
                this.genome=this.genomeDaughter.clone();
            }
        }
    }


    public void enterApoptosis(){
        // Eventualmente isto poderá ser feito no Model...
        modelo.cells.remove(this);
        modelo.cellSpace.putObjectAt(this.x,this.y,null);
//          System.out.println("Killed "+this.who );
    }
    public void induceAngiogenesis(){
        //XXX
        Vascular vi = null;
        double distv = modelo.getWorldXSize()*2;
        double dist = 0.0;
        for (int i = 0; i < modelo.vascular.size(); i++) {
            Vascular v = (Vascular)modelo.vascular.get(i);
            dist = Math.sqrt(Math.pow((v.getX()-this.x),2)+Math.pow((v.getY()-this.y),
2));
            if (dist<distv){
                distv=dist;
                vi=v;
            }
        }
        vi.callVascular(this);

//          System.out.println(dist);
    }
    public boolean isCancer(Cell c){
        int soma=0;
        boolean out=false;
        for (int i = 0; i < c.genome.length; i++) {
            soma+=c.genome[i];
        }
        if (soma==c.genome.length){
            out=true;
        }
        return out;
    }
    public void setBoleanGenes(){
        if (genome[0]==1){
            this.growthSignal=true;
        }
        if (genome[1]==1){
            this.sustaindedAgiogenisi=true;
        }
        if (genome[2]==1){
            this.ignoreGrowthInhibit=true;
        }
        if (genome[3]==1){
            this.evadeApoptosis=true;
        }
        if (genome[4]==1){
            this.limitlessReplication=true;
        }
        if (genome[5]==1){
            this.geneticInstability=true;
        }
    }
    /*  Ideias para o Step de Vida de Cada Célula...
     *
     *  O sistema Vascular poderá ser o responsável por colocar a quantidade de
     *  nutrientes certa em cada célula.
     *  Para isso cria-se um espaço para Nutrientes...
     *  E numa vizinhaça de Moore X faz-se a actualização... dos níveis de nutrientes
     *  até um nível considerado máximo... o decaimento será exponêncial por
     *  exemplo colocando o nível a metade...
     *  é preciso verificar se no nível
     */
    /*  Setter And Getters
     */

    public int getFase() {
        return fase;
    }

    public int getGenome() {
        int soma = 0;
        for (int i = 0; i < genome.length; i++) {
            soma+=genome[i];
        }
        return soma;
    }

    public boolean isEvadeApoptosis() {
        return evadeApoptosis;
    }

    public boolean isGeneticInstability() {
        return geneticInstability;
    }

    public boolean isGrowthSignal() {
        return growthSignal;
    }

    public boolean isIgnoreGrowthInhibit() {
        return ignoreGrowthInhibit;
    }

    public boolean isLimitlessReplication() {
        return limitlessReplication;
    }

    public boolean isSustaindedAgiogenisi() {
        return sustaindedAgiogenisi;
    }

    public int getTelomeres() {
        return telomeres;
    }

    public String getWho() {
        return who;
    }

}
```

```java
/*
 * Acid.java
 *
 * Created on May 28, 2007, 4:20 PM
 *
 */

package cellcom;

import java.awt.Color;
import uchicago.src.sim.gui.Drawable;
import uchicago.src.sim.gui.SimGraphics;

/**
 * Class for the Acid Space
 * @author David Rodrigues
 */
public class Acid implements Drawable {
    int x;
    int y;
    private Model modelo;
    private double concH;
    private double nextConcH;
    private Color cor;

    /**
     * Creates a new instance of Acid
     * @param modelo Reference to the model this class is in.
     */
    public Acid(Model modelo)  {
        this.modelo=modelo;
    }


    /**
     * Sets the XX and YY coordinates of the Class.
     * @param i X coord.
     * @param j Y coord.
     */
    public void setXY(int i, int j) {
        this.x=i;
        this.y=j;
    }

    /**
     * Draw method called by repast to make the representatio of the class in the
display
     * @param g SimGraphics object that will draw this class in the Display
     */
    public void draw(SimGraphics g) {
        if (getPH()<6.0){
            this.cor=new Color(100,168,242,0);
        } else if (getPH()>7.4){
            this.cor=new Color(100,168,242,127);
        } else {
            this.cor=new Color(100,168,242,(int)((getPH()-6.0)*90));
        }


        g.drawFastRect(cor);
    }

    /**
     * Getter for the Acid XX coordinate
     * @return XX coord.
     */
    public int getX() {
        return this.x;
    }

    /**
     * Getter for the YY coordinate for this Acid Class
     * @return YY Coord.
     */
    public int getY() {
        return this.y;
    }
    /**
     * Calculates and returns the pH at this Acid Class X,Y
     * @return pH
     */
    public double getPH(){
        return -1*Math.log10(this.concH);
    }
    /**
     * Set the concentration of acid in this lattice location
     * @param pH pH
     */
    public void setPH(double pH){
        this.concH=Math.pow(10, -1*pH);
    }

    /**
     * gets the concentration of Acid
     * @return Acid conentration in mol/L
     */
    public double getConcH() {
        return concH;
    }

    /**
     * Sets Acid Concentration
     * @param concH Acid Concentration in mol/L
     */
    public void setConcH(double concH) {
        this.concH = concH;
    }

    /**
     * Set the concentration of Acid for the diffusion process
     * @param nextConcH Acid concentration in mol/L
     */
    public void setNextConcH(double nextConcH) {
        this.nextConcH = nextConcH;
    }

    /**
     * Getter for the Acid concentration in the diffusion mechanism
     * @return Acid Concemtration in mol/L
     */
    public double getNextConcH() {
        return nextConcH;
    }

}

```

```java
/*
 * MoleculeSpace.java
 *
 * Created on May 21, 2007, 10:02 PM
 *
 */

package cellcom;

import java.awt.Color;
import uchicago.src.sim.gui.Drawable;
import uchicago.src.sim.gui.SimGraphics;

/**
 * Class that holds the methods to draw the nutrients Space
 * @author David Rodrigues
 */
public class MoleculeSpace implements Drawable {

    // Melcular Space Will Hold quantities that need to be avaiable to all cells
    /**
     * Nutrients concentration
     */
    public double nutrients=0.0;
    /**
     * Growth Factor Concentration
     */
    public double cellGrowhtFactor=0.0;
    /**
     * Vascular Growth Factor Concentration
     */
    public double vascularGrowthFactor=0.0;
    /**
     * Acid Concentration
     */
    public double concHydrogen=0.0;
    /**
     * Growth Inhibitors Concentration
     */
    public double inhibitors=0.0;
    /**
     * X position on the lattice
     */
    public int x;
    /**
     * Y position in the lattice
     */
    public int y;
    private Model modelo;
    private String who="Nutrients";

    /**
     * Creates a new instance of MoleculeSpace
     * @param modelo Reference to the model that holds this class.
     */
    public MoleculeSpace(Model modelo) {
        this.modelo=modelo;
    }

    /**
     * Set the X and Y coordinte for this class
     * @param i X coord.
     * @param j Y Coord.
     */
    public void setXY(int i, int j) {
        this.x=i;
        this.y=j;
    }

    /**
     * Method to draw the nutrients in the display
     * @param g SimGraphics object that defines how this class is drawn in the display
     */
    public void draw(SimGraphics g) {
        float nut = (float)(((Double)modelo.nutrienteSpace.getValueAt(this.x,this.y)).
doubleValue());
```

```
        nut = 2*nut / (float)modelo.getVascularNutrients();
        if (nut > 1){
            nut=1;
        }
        Color cor = new Color(nut / 2, nut, nut / 2, (float)0.5);
        g.drawFastRect(cor);
    }

    /**
     * Getter for the X position
     * @return X Coord.
     */
    public int getX() {
        return this.x;
    }

    /**
     * Getter for the Y position of this class
     * @return Y coord.
     */
    public int getY() {
        return this.y;
    }

    /**
     * Unique Id of this class
     * @return ID of the class
     */
    public String getWho() {
        return who;
    }

    /**
     * Getter for the nutrients concentrations
     * @return Nutrients concentration
     */
    public double getNutrients() {
        return ((Double)modelo.nutrienteSpace.getValueAt(this.x,this.y)).doubleValue();
    }

}
```

```java
/*
 * Vascular.java
 *
 * Created on May 18, 2007, 4:52 PM
 *
 */

package cellcom;

import java.awt.Color;
import java.util.Vector;
import uchicago.src.sim.gui.Drawable;
import uchicago.src.sim.gui.SimGraphics;

/**
 * Vascular System Class
 * @author David Rodrigues
 */
public class Vascular implements Drawable {
    private int x;
    private int y;
    private Color cor;
    private Model modelo;
    private String who="Vascular ";

    /**
     * Creates a new instance of Vascular
     * @param modelo Reference to the model that is running
     */
    public Vascular(Model modelo) {
        this.cor = Color.RED;
        this.modelo=modelo;
        this.who+=modelo.vascularNumber++;
    }
    /**
     * Creates a new Vascular in the direction of the Calling Cell
     * @param c Calling Cell
     */
    public void callVascular(Cell c){
//        System.out.println("callVascular");
        double dx = c.getX() - this.x;
        double dy = c.getY() - this.y;
        double teta= Math.atan2(dy,dx);
        double ang = Math.PI / 8;
        int nx=0;
        int ny=0;
        if (teta > 5 * ang || teta < -5 * ang ){
            nx=-1;
        }
        if (teta < 3 * ang && teta > -3 * ang){
            nx=+1;
        }
        if (teta > -7 * ang && teta < -1 * ang ){
            ny=-1;
        }
        if (teta < 7 * ang && teta > 1 * ang ){
            ny=+1;
        }
//        System.out.println(nx+","+ny);
        Vascular b = new Vascular(modelo);
        if (modelo.vascularSpace.getObjectAt(this.x+nx,this.y+ny)==null &&
                modelo.vascularSpace.getMooreNeighbors(this.x+nx,this.y+ny,false).size()
<=1){

            b.setXY(this.x+nx, this.y+ny);

            modelo.vascularSpace.putObjectAt(this.x+nx,this.y+ny,b);
            modelo.vascular.add(b);
            //b.diffuseNutrients();
//            modelo.diffuseNutrients();
        }



//        System.out.println(teta);
    }

    /**
     * repast method to draw the Vascular system
     * @param g Defines the type of graph to draw in the display for the vascular system.
     */
    public void draw(SimGraphics g) {

        g.drawHollowFastOval(this.cor);
        cor=new Color(255,0,0,170);

        g.drawFastRect(cor);

    }

    /**
     * Getter for the XX coordinates of this Vascular Cell in the Vascular Space
     * @return XX coordinates
     */
    public int getX() {
        return this.x;
    }

    /**
     * Getter for the YY coordinates of this Vascular Cell in the Vascular Space
     * @return YY coordinates
     */
    public int getY() {
        return this.y;
    }
    /**
     * Sets the XX and YY coordinates for the Vascular Cell
     * @param x X coord.
     * @param y Y coord.
     */
    public void setXY(int x, int y){
        this.x=x;
        this.y=y;
    }
    /**
     * Set a value for the nutrients in a determined nutrient space
     * @param nut The value to put in this cell
     */
    public void diffuseNutrients(double nut){
        modelo.nutrienteSpace.putValueAt(this.x,this.y,nut);
    }

    /**
     * Getter for the id of the Vascular cell
     * @return The ID of the Vascular Cell
     */
    public String getWho() {
        return who;
    }

}

```