Single-Cell-Based Models in Biology and Medicine
ed. by A.R.A. Anderson, M.A.J. Chaplain, K.A. Rejniak
Mathematics and Biosciences in Interaction, 151–167
© 2007 Birkhäuser Verlag Basel/Switzerland

II. 4 The Glazier–Graner–Hogeweg Model: Extensions, Future Directions, and Opportunities for Further Study

Ariel Balter, Roeland M. H. Merks, Nikodem J. Popławski, Maciej Swat and James A. Glazier

Abstract. One of the reasons for the enormous success of the Glazier–Graner–HogewegGlazier–Graner–Hogeweg Model (*GGH*) model is that it is a framework for model building rather than a specific biological model. Thus new ideas constantly emerge for ways to extend it to describe new biological (and non-biological) phenomena. The GGH model automatically integrates extensions with the whole body of prior GGH work, a flexibility which makes it unusually simple and rewarding to work with. In this chapter we discuss some possible future directions to extend GGH modeling. We discuss *off-lattice* extensions to the GGH model, which can treat fluids and solids, new classes of model objects, approaches to increasing computational efficiency, parallelization, and new model-development platforms that will accelerate our ability to generate successful models. We also discuss a non-GGH, but GGH-inspired, model of plant development by Merks and collaborators, which uses the Hamiltonian and Monte-Carlo approaches of the GGH but solves them using Finite Element (*FE*) methods.

1. Off-Lattice Extensions to the GGH Model

While the use of a fixed lattice makes GGH computations very simple compared to FE methods, we have seen that this simplicity comes with certain costs (see chapter II.1, section 6). The inclusion of the *g*eneralized cell as an independent entity in the GGH allows the definition of *i*nternal cell states tied to the cell rather than the lattice. These internal cell states allows us to specify interactions that are calculated either partially or entirely *off-lattice*, *i.e.* they are not tied to the grid structure of the cell lattice or any external field lattices (as in section 1.2.2). The advantage of this addition is that it may overcome potential negative effects of lattice discretization without abandoning the speed and simplicity of a lattice-based formalism. The main costs of moving off lattice are increased memory usage and slower computation.

The original CPM included simple off-lattice concepts such as cell volumes and surface areas, but these quantities still had a granularity of one lattice site. An example of a strictly off-lattice quantity is a cell's *center of mass*:

$$\vec{cm}(\sigma) = v^{-1}(\sigma) \sum_{\vec{i} \ni \sigma'(\vec{i})=\sigma} \vec{i}, \tag{1}$$

where $v(\sigma)$ is the volume of cell σ and $\vec{i}$ is a lattice site vector. We can form new off-lattice entities from existing ones. For example, by tracking a cell's center of mass from one time step to the next, we can estimate its *velocity*:

$$\vec{vel}(\sigma, t) = \frac{\vec{cm}(\sigma, t) - \vec{cm}(\sigma, t - \Delta t)}{\Delta t}. \tag{2}$$

Many of the current and imminent extensions of the GGH model employ off-lattice extensions. We will discuss several such extensions: implementation of inertia, viscous fluid flow and advective diffusion, cell shape and polarity control and rigid body motion. This use of off-lattice extensions to the GGH is completely different from a switch to entirely *lattice-free methods*, which omit the underlying lattice entirely in favor of FE-implementations of a GGH-like Hamiltonians. We will show one such implementation to illustrate this fundamental difference.

1.1. Persistent Cell Movement—Inertial Constraints

Normal motion in the GGH model is Aristotelian. Velocity is proportional to applied force, with no inertia. While this dynamics is often appropriate in a biological context, we also encounter situations in which motion is inertial, *e.g.*, in fluid flow or the motion of large objects, or persistent, *e.g.*, in cell migration (because cells take time to reorganize their internal machinery to change their direction of motion) [24]. Somewhat surprisingly, we can model both inertia and cytoskeletal persistence time by adding to our Hamiltonian an *inertial term* of form:

$$\mathcal{H}_{\text{inertia}}(\Delta t) = \sum_{\sigma} \lambda_{\text{inertia}}(\sigma) ||\vec{vel}(\sigma, t) - \vec{vel}(\sigma, t - \Delta t)||^2, \tag{3}$$

where $\vec{vel}(\sigma, t)$ is the instantaneous center-of-mass velocity of cell σ and λ_{inertia} controls the persistence time. Typically, Δt is one or more MCS. If $\lambda_{\text{inertia}} = 0$, objects undergo uncorrelated Brownian motion. If λ_{inertia} is large, motion is ballistic. We are only beginning to explore the possibilities that an inertial term presents in GGH modeling and its relation to more traditional Newtonian formulations of inertia. It is worth noting that cell orientation (chapter II.2, section 7.2) and cell elongation [18] induce persistent cell motion.

1.2. Fluid Dynamics

Surprisingly, given that the GGH model does not have a concept of a solid object, it also lacks two features key to real fluids, viscous dissipation due to shear and advection of chemical fields (which are normally tied to an auxiliary lattice in GGH simulations – see chapter II.1, section 7.3). Off-lattice extensions allow us to recover both of these important behaviors.

1.2.1. Viscous flow. We can introduce viscosity in a fluid using a *relative velocity constraint* between generalized cells. For an incompressible fluid, we add to the Hamiltonian a viscous term [5]:

$$\begin{aligned}\mathcal{H}_{\text{visc}} = \lambda_{\text{visc}} \sum_{(\sigma,\sigma')\text{ neighbors}} & S(\sigma,\sigma') \frac{(vel_x(\sigma) - vel_x(\sigma'))^2}{||\vec{cm}(\sigma) - \vec{cm}(\sigma')||^2} \\ \times & \sqrt{\frac{(cm_y(\sigma) - cm_y(\sigma'))^2 + (cm_z(\sigma) - cm_z(\sigma'))^2}{||\vec{cm}(\sigma) - \vec{cm}(\sigma')||^2}} \\ + & \text{ cyclic permutation of } (x, y, z), \end{aligned} \tag{4}$$

where $||\ ||$ is the Euclidean norm, subscripts indicate directional components, and $S(\sigma, \sigma')$ is the common contact area between cells σ and σ'. The parameter λ_{visc} corresponds to the Navier–Stokes viscosity η. The GGH model, with $\mathcal{H}_{\text{visc}}$ added to the Hamiltonian, successfully reproduces Poiseuille flow and, with the advection diffusion of section 1.2.2, it reproduces Taylor–Aris dispersion [5]. No one has yet tried to combine inertial, Eq.(3) and viscous terms to solve the full Navier–Stokes equations using the GGH method. Such a fluid solver would probably be inefficient, especially at high Reynolds numbers, but it would be extremely convenient, and would allow modeling of situations like blood flow and biofilm growth in a flowing fluid, where inertia and transport are crucial. This method may also work for complex fluids, *e.g.*, in blood rheology.

1.2.2. Advective diffusion. One problem with recording the concentration of chemicals in external lattices is that they do not respond to flow occurring in the cell lattice. Moving biological cells or fluid particles should carry with them, or *advect* chemical fields. In the GGH model, chemicals remain fixed to the external lattice. Dan *et al.* implemented advection by allowing diffusion directly between generalized cells, fluid particles, or subcells, which carry all diffusing chemicals [5], abandoning external lattices and lattice diffusion solvers. Diffusion uses the forward-Euler method (chapter II.1, section 7.3.2), but between neighboring cells rather than lattice sites. If we like, we can weight by the inverse distance between centers of mass of neighboring cells, or by contact areas between neighboring cells. As with normal lattice diffusion (see chapter II.1, section 7.3.2), we can include spatially-varying diffusion constants, secretion and decay rates and reaction terms. One disadvantage of this method is that it coarsens our description of chemical fields so that we only resolve concentrations at the scale of individual cells.

1.3. Cell Polarity

Cells are often highly asymmetric, with their cytoskeleton primarily determining and maintaining the asymmetry. Cells in epithelial sheets have distinct apical, lateral and basal surfaces, and migrating cells have a leading and trailing surface. The basal–apical or trailing-surface–leading-surface vector usually defines a primary cell orientation, or *polarity*, but additional orientations may also be important. *E.g.*, in-plane polarization (via the planar polarity path) helps epithelial cells to form long-range patterns in response to external signals. In addition, cells like neurons can have extremely complex structures. Since the cells of the basic GGH model are isotropic, representing even the simplest aspects of cell polarity and shape requires extensions. The off-lattice extensions needed to give structure to cells also permit the creation of movable elastic solids and visco-elastic materials with well-defined behaviors, both classes of material lacking in the basic GGH model.

1.3.1. Cell orientation. To track cell polarity, we can assign each cell an *orientation vector* as an internal state. Making all cell properties, *e.g.*, chemotaxis and adhesion, depend on this orientation is then straightforward. However, determining how the orientation vector should evolve is not obvious. Cell elongation (see section 1.3.2) creates polarized cells in which the orientation passively follows the cell geometry.

The simplest way to describe highly complex cell shapes and variations in properties within the cell is to use compartmental cells (section 1.3.3). Different compartments can respond to external stimuli (contact with certain cell types, presence of external fields, *etc.*) differently, causing them to orient in their surroundings. However, compartmentalization is computationally expensive. The optimal strategy will depend on the phenomena being modeled, and all methodologies require further study. One current off-lattice method associates an orientation vector with each cell (chapter II.2, section 7.2).

1.3.2. Inertia-tensor cell-shape constraints. Zajac used a constraint on the eigenvalues of cells' 2nd moment tensor, or *inertia tensor*, to control cell shape in a model of convergent extension [28]. Later, Merks used it to control cell elongation in a vasculogenesis model [20]. Define $I(\sigma)$, the inertia tensor of cell σ:

$$I_{\alpha\beta} = \sum_{\vec{i}\ni\sigma'(\vec{i})=\sigma} \left(i_\alpha i_\beta \delta(\alpha,\beta) - \frac{1}{v_{\rm t}(\sigma)} \sum_{\vec{j}\ni\sigma'(\vec{j})=\sigma} i_\alpha j_\beta \right), \tag{5}$$

where α and β denote directional indices.

The inertia tensor translates any object into an equivalent ellipsoid. Let $I_1 > I_2\,[> I_3]$ be the eigenvalues of I. If cells are roughly ellipsoidal, the eigenvalues of the inertia tensor give the ratios of the lengths of their principal axes. Scaling these ratios by the volume of the cell gives us the lengths. We can now constrain cells to maintain a given length or shape. For instance, in 2D, we could impose a length constraint [20]:

$$\mathcal{H}_{\rm l} = \sum_\sigma \lambda_{\rm l} \left(l(\sigma) - L_{\rm t}(\sigma) \right)^2, \tag{6}$$

where $l(\sigma)$ is the length along the long axis of the cell σ and $L_{\rm t}(\sigma)$ the target length. Alternatively, we could constrain the aspect ratio, $r = I_1/I_2$, where I_1 and I_2 are the principle axes:

$$\mathcal{H}_{\rm r} = \sum_\sigma \lambda_{\rm r} \left(r(\sigma) - R_{\rm t}(\sigma) \right)^2, \tag{7}$$

where $R_{\rm t}$ is the target ratio.

To prevent cells from splitting because of the shape constraint, Merks *et al.* introduced an approximate *local connectivity constraint* in 2D that penalizes cells which become multiply-connected during an index-copy attempt [20]. To check whether an index copy into site $\vec{i}$ changes local connectivity, they counted how many neighbors $\vec{j}_n$ of site $\vec{i}$ have the same index as $\vec{i}$, while the next lattice site clockwise around $\vec{i}$ has an index different from that of $\vec{i}$. If the quantity:

$$\sum_n \delta(\sigma(\vec{i}), \sigma(\vec{j}_n))(2 - \delta(\sigma(\vec{i}), \sigma(\vec{j}_{n+1})) - \delta(\sigma(\vec{i}), \sigma(\vec{j}_{n-1}))) > 2 \tag{8}$$

(with the sum running in cyclic order), and the local neighborhood contains more than two non-medium cells, changing the index at the site destroys the local connectivity. They then made cell fragmentation energetically costly by assigning a large energy penalty to updates that change local connectivity. No similar local rule has been developed in three dimensions (*3D*). Such an algorithm would be extremely useful. Inertia-tensor description of cell elongation can also be used to measure elastic strain and control mitotic rate in tumor growth (chapter II.2, section 4.2).

1.3.3. Compartmental cells. For his *myxobacteria* simulations, Andreas Deutsch modeled elongated bacteria that are able to bend, but not too much [1, 27]. He subdivided individual bacteria into strings of subcellular domains, where $m(\sigma)$ is the number of subcells in cell σ. He then defined contact energies between subdomains within a cell to give the cell a particular geometry. *E.g.*, high contact energy between the front and rear subcells prevents a bacterium from forming a loop.

To control side-to-side undulations, Deutsch introduced a *bending energy* from polymer physics. The elastic bending energy of a rod is proportional to its mean-squared curvature. In Deutsch's simulations, every three subcells define a local curvature. If we define $\vec{cm}(\sigma, \mu)$ to be the center of mass of the μ^{th} subdomain of cell σ, then the *local radius of curvature* is:

$$\begin{aligned} R_{\text{curve}}(\sigma, \mu) &= ||\vec{cm}(\sigma, \mu+1) - \vec{cm}(\sigma, \mu-1)|| \times [||\vec{cm}(\sigma, \mu+1) - \vec{cm}(\sigma, \mu)||^2 \\ &+ ||\vec{cm}(\sigma, \mu) - \vec{cm}(\sigma, \mu-1)||^2 - ||\vec{cm}(\sigma, \mu+1) - \vec{cm}(\sigma, \mu-1)||^2]^{-1}. \end{aligned} \tag{9}$$

Since the *mean curvature* $\kappa = \frac{1}{R_{\text{curve}}}$, and for a rod, $\mathcal{H}_{\text{bend}} = \int \lambda_{\text{bend}}(l)\kappa^2 dl$, where λ_{bend} is the *bending modulus* of the rod, the equivalent *bending energy* in the GGH model is:

$$\mathcal{H}_{\text{bend}} = \sum_\sigma \sum_{\mu=1}^{m(\sigma)-2} \lambda_{\text{bend}}(\sigma, \mu) R^{-2}_{\text{curve}}(\sigma, \mu). \tag{10}$$

Thus, the term resists bending of the bacterium.

Another approach to limiting bending is to constrain the angles between the centers of mass of the subcells:

$$\mathcal{H}_{\text{angle}} = \sum_\sigma \sum_{\mu=2}^{m(\sigma)-1} \lambda_{\text{angle}}(\sigma, \mu)(\Theta(\sigma, \mu, \mu-1) - \Theta_{\text{t}}(\sigma, \mu, \mu-1))^2, \tag{11}$$

where $\Theta(\sigma, \mu, \mu{-}1)$ is the angle between $\vec{cm}(\sigma, \mu)$ and $\vec{cm}(\sigma, \mu{-}1)$. This constraint places rotational springs at the joints between the lines connecting the subcells' centers of mass. Deutsch maintained the overall length of the cell by treating the lines connecting the centers of mass as springs with an equilibrium length $L_{\rm t}(\sigma,\mu)$ and a spring constant $\lambda_{\rm compression}$:

$$\mathcal{H}_{\rm compression} = \sum_{\sigma} \sum_{\mu=1}^{\mu(\sigma)-1} \lambda_{\rm compression}(\sigma,\mu)(||\vec{cm}(\sigma,\mu) - \vec{cm}(\sigma,\mu-1)|| - L_{\rm t}(\sigma,\mu))^2. \quad (12)$$

1.4. Elastic Solids

All cell-based materials in the classical GGH model are intrinsically visco-elastic. The model has no elastic solids. We can implement elastic and plastic solids using a simple algorithm similar to that in Eq.(12). Take any object σ and divide it into $m(\sigma)$ subcells μ, where each subcell contacts at least two other subcells of σ and the triangulation defined by the connections between the centers of mass $(\vec{cm}(\sigma,\mu))$ of neighboring subcells is topologically rigid. Then:

$$\mathcal{H}_{\rm elastic} = \sum_{\sigma} \sum_{\mu,\nu=1\ (\mu,\nu\ \text{neighbors})}^{m(\sigma)} \lambda_{\rm rigid}(\sigma,\mu,\nu)(||\vec{cm}(\sigma,\mu) - \vec{cm}(\sigma,\nu)|| - L_{\rm t}(\sigma,\mu,\nu))^2, \quad (13)$$

where $L_{\rm t}(\sigma,\mu,\nu)$ is the target length of the link between subcells μ and ν inside cell σ, and $\lambda_{\rm rigid}$ is the *Young's modulus* of the material, which defines the total stress energy of σ. If we define neighborhood by distance between centers of mass rather than adjacency, then the resulting cell is visco-elasto-plastic. For instance, we could include only contacting subcells, or all subcells within a cutoff distance. Once two subcells separate further than the cutoff, they no longer interact elastically, allowing slow, plastic creep.

1.5. A Lattice-Free, Finite Element, GGH-Inspired Model of Plant Development

To illustrate the difference between the off-lattice extensions of the GGH, which we have been discussing, and lattice-free methods, we present a FE model developed by Merks and collaborators, which uses a GGH-inspired Hamiltonian but makes all of its computations using a FE mesh.

Most developmental phenomena simulated with the GGH model are examples of *plastic* morphogenesis. Since the constituent cells swarm, mix, sort, or aggregate, the resulting tissues behave as living "clays" in which biological form and pattern arise primarily through cell motility. In plants and some animal tissues (*e.g.*, in *Drosophila* epithelia, see [8]) the relative positions of the cells are effectively fixed, and only cell division changes tissue shape. Plant cells cannot migrate and, with very few exceptions, do not slide past each other. Consequently, plant morphogenesis depends on patterned cell division and cell expansion, instead of cell migration and tissue folding. This mode of development is often called *symplastic growth* [6, 23].

Although GGH modeling of symplastic development is possible, it requires computationally-expensive topology constraints or finely-tuned adhesion energies to suppress cell motility and surface fluctuations, precisely the phenomena that make the GGH model suitable for simulating animal development. Instead, Merks and collaborators developed a FE, lattice-free, GGH-inspired model that explicitly excludes plastic cell movement.

As we discussed in chapter II.1, section 2.1, the energy-minimization philosophy of the GGH model does not require lattice-based Monte-Carlo solution methods. Instead, we use two-dimensional (*2D*) lattice-free, polygonal finite elements to represent cell walls in a tissue [21, 25]. However, we retain a Monte-Carlo-based energy minimization dynamics that allows straightforward modeling of cell behaviors, including expansion, division, and active shape change. We allow cell walls to move according to rules derived from the classic equations for cell expansion [16]:

1. The intracellular turgor pressure exerts a uniform force on cell walls, attempting to enlarge cells.
2. The elastic cell walls counteract the turgor pressures.
3. Walls expand irreversibly if stretched over a threshold, via *cell-wall yielding.*

As in GGH-models, we define a target area A_{t} at which the cell's turgor pressure balances the ambient pressure, and a target length L_{t} for each wall element. We denote the *actual* cell area, a, and wall-element length, l. We can then describe the balance between turgor pressure and cell wall resistance in terms of a *generalized energy* or *Hamiltonian*:

$$\mathcal{H} = \sum_i (a(i) - A_{\mathrm{t}}(i))^2 + \sum_j (l(j) - L_{\mathrm{t}}(j))^2, \tag{14}$$

where indices i and j sum over all cells and polygon edges, respectively. We can add additional terms to the Hamiltonian, *e.g.*, body forces like gravity, or cell-length constraints.

The simulation uses a Metropolis-like dynamics (chapter II.1, section 2.3.2). We iteratively choose a random node, and attempt to move it in a random direction $\vec{x}_{\mathrm{new}} = \vec{x}_x + \xi\vec{r}$, where $\vec{r} = \{\rho, \theta\}$ is a random vector chosen uniformly within the unit circle (*i.e.* $\rho \in [0, 1]$ and $\theta \in [0, 2\pi)$) and ξ is the step size. To model cell wall yielding, we introduce new nodes whenever a polygon edge's length exceeds a threshold value. Since we assume shape relaxation is fast compared to biological changes, we make biologically motivated changes, like turgor pressure increases, cell divisions or active cell shape changes, only after turgor pressures, cell expansion and any resulting cell displacements have equilibrated.

We are currently using this method to model leaf morphogenesis, which involves a dynamic interplay between leaf venation, cell expansion and cell division. Fig.1 shows a sample simulation, in which a set of ordinary differential equations (*ODE*s) in each cell implements Meinhardt's four-species reaction-diffusion model as a prototype for leaf venation [17]. See also a simulation MovII.4.1 from the accompanying DVD. Cell expansion, which we model as a gradual increase in cells target areas (Chapter II.1, section 7.2.1), is fastest where concentrations of a diffusing growth repressor are lowest, near the developing vasculature which drains it, while the vasculature itself expands four times more slowly than the surrounding tissue. Cells divide along their shortest axis when they reach twice their original

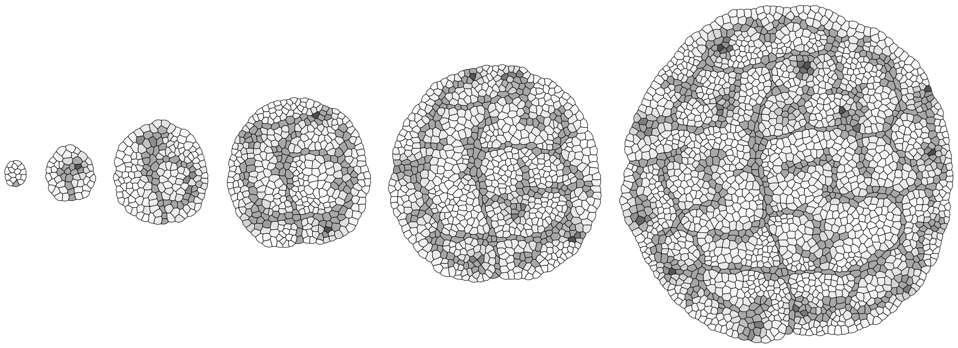

FIGURE 1. Relation between leaf growth and venation patterning in a symplastic growth model based on [17]. Dark grey cells are vascular. Light grey cells are normal leaf cells, with the degree of shading indicating the local concentration of the activator which regulates cell growth. See also a simulation MovII.4.1.

size (Chapter II.1, section 7.2.2). We are currently developing more realistic models of leaf venation patterning, which include several molecules known to regulate leaf venation, including the phytohormone auxin and its transporter protein PIN1. Eventually, the set of ODEs in each cell will describe the production, decay, regulation, cellular localization and transport of the molecules regulating leaf venation patterning.

2. Modeling Extracellular Matrix

Current GGH simulations either represent Extracellular Matrix (*ECM*) as a special cell, often the default cell index on lattice sites not occupied by any other cell, producing an ECM which behaves as an infinitely compressible, inviscid fluid, or as a non-diffusing chemical field which is non-deformable. These approximations may be adequate in many cases. However, in many morphogenetic processes (such as bone formation) ECM properties are crucial and require more realistic representation.

Although ECM tends to receive less attention than cells (which seem to do all the work during morphogenesis), it is a complex, varied and highly structured material, which includes networks of insoluble protein *fibrils* and numerous soluble components. *Collagens* and *elastins* are usually the two primary classes of structural proteins contributing to the fibrils. Soluble polymers, primarily long carbohydrate chains which bind water, link to these proteins. Rigid ECMs are crucial for maintaining tissue shape, and form the structural portion of most connective tissues and bones. ECM structure is heterogeneous, with important patterns at length-scales from tens of angstroms to millimeters, making it extremely difficult to model. Its fiber structure gives ECM highly anisotropic mechanical properties, including visco-elasto-plasticity, and often provides crucial directional cues for cell migration.

ECM also mediates signaling between cells. ECM often binds cytokines and growth factors, acting as a chemical reservoir and greatly reducing the effective diffusion constant of diffusible signaling molecules. Ligands in the ECM also interact with specific cell-surface receptors (*integrins*) and cells can sense variations in ECM rigidity and fiber orientation. The ECM records the history of cell migration and differentiation during morphogenesis in its fibril pattern, which then guides further morphogenesis. Differences in ECM architecture account for much of the difference between normal and scar tissue. Appropriate synthetic ECM can induce tissue regrowth in adults, while metastasis occurs when tumor cells lose their binding to ECM.

Existing GGH methods can describe some ECM properties. For instance, we can model ECM binding of cytokines by lowering the diffusion constant of these chemicals in regions of ECM. However, many ECM properties, *e.g.*, visco-elasto-plasticity (see section 1.4) and fibril orientation (see section 2), call for a new *vector-field* class of object.[1]

Describing fiber orientation in terms of a vector field is complex because, at the typical scale of a GGH lattice, each lattice site will contain fibers of almost every possible orientation, requiring an impossibly complex description. A practical description of ECM requires careful optimization of the trade-offs among the number of allowed fiber orientations and fiber types per site and the angular resolution of those orientations. Options could range from a simple fiber with arbitrary orientation at each site, to a dozen fibers, each with variable concentration but only eight allowed orientations each at each site. The passage of cells can then change the fiber structure, which the field represents. We can model the effect of fiber orientation on morphogenesis by including terms in the Hamiltonian that couple ECM vector fields to cell chemotaxis, haptotaxis (chapter II.1, section 7.3), or cell polarization (section 1.3).

3. Improving Computational Efficiency Using Alternative Evolution Algorithms

Because it is extremely simple to implement, the modified-Metropolis algorithm has been a central part of GGH modeling. However, the modified-Metropolis algorithm rejects most index-copy attempts, making the algorithm inefficient and its stochasticity makes parallelization awkward. We also need to improve our diffusion

[1]Vector fields could provide an alternative method to implement fluid flow. Instead of using the fluid-particle method we described in section 1.2.1, we could define a flow field in the ECM and use a sophisticated Navier–Stokes-equation solver to update the fluid flow. Cell motion would impart vector boundary conditions and the simulation would become an immersed-boundary calculation.

solvers, since many current simulations spend only 10% of their time on GGH solutions and 90% solving the diffusion equation. Fortunately, diffusion algorithms, even Euler diffusion, are synchronous, and hence parallelize easily.

3.1. Rejection-Free Dynamics

Of the non-Metropolis Monte-Carlo algorithms, *rejection-free* algorithms such as the *N-fold way* and *kinetic Monte-Carlo* are particularly attractive [2, 9, 7, 15]. Rather than considering a trial index copy at each step, which we may or may not accept, these methods choose among allowed lattice updates at each step. Thus, while they spend much more time computing possible updates, they save the time normal Metropolis spends on rejected updates. The computational gain depends on the average number of possible updates, the fraction of possible updates that need to be recalculated after each update, and the average acceptance rate. If the product of these three numbers is less than one, rejection-free algorithms are faster. In some non-GGH cases, the gains can be dramatic (rejection-free code can be thousands, or even millions of times faster). Rejection-free methods can also facilitate parallel computing (section 4.2).

When we know the transition probabilities for a continuous-time Markov process, we can construct a discrete-time process or *embedded Markov chain* as well as a distribution of transition times. Consider the continuous-time Markov process with transition rates $w(j \to i) = \lim \underset{\Delta t \to 0}{\longrightarrow} p(i, t + \Delta t | j, t)$,

$$\frac{dp\,(i,t)}{dt} = \sum_j w\,(j \to i) p(j,t). \tag{15}$$

We can use Eq.(15) to calculate the probability for the transition $j \to i$ *given* that *some* transition has occurred:

$$f\,(j \to i) = \frac{w(j \to i)}{\sum\limits_{k \neq j} w(j \to k)}. \tag{16}$$

Eq.(16) is the jump probability for the discrete-time *master equation*:

$$p\,(i, n+1) = \sum_j f(j \to i) p(j,n), \tag{17}$$

where the index n measures how many transitions have occurred rather than the amount of time that has passed. Since the stochastic process underlying the master equation is memoryless, the waiting time to leave state j is exponentially distributed, with a rate parameter:

$$\lambda_j = \sum_{k \neq j} f\,(j \to k). \tag{18}$$

If the transition rates between states are proportional to $\exp[-\Delta\mathcal{H}/T]$ (Boltzmann weighting), where $\Delta\mathcal{H}$ is the difference in the states' energies, we can use the following rejection-free Monte-Carlo algorithm:

1. Determine all allowable lattice updates from the current lattice configuration.
2. Calculate the energy changes between them.

3. Calculate the time parameter for each of these transitions, λ_k.
4. Choose a random time parameter, λ_{random}, between 0 and $\max\{\lambda_k\}$.
5. Find the transition n for which $|\lambda_k - \lambda_{random}|$ is the smallest, and update the lattice using this transition.
6. Increment the time by a random time interval chosen with probability: $p(t)dt = \lambda_n \exp[-\lambda_n t]dt$.
7. Go to 1.

We may be able to bypass step 6 and simply increment the time by $1/\lambda_n$. Determining whether this approach works on average would be a nice research project. Because most of the allowed updates and their energy changes will *not* change after a given update, the complete set of possible updates needs to be evaluated only once at the beginning of a simulation, with much faster local updates thereafter. The longer the effective interaction range in the model, the larger the number of recalculations necessary after each step and the less efficient the algorithm. Thus, the gains of using rejection-free method are greatest for a simple, nearest-neighbor Ising model, while rejection-free algorithms are much worse than Metropolis for infinite-range interactions.

3.2. The Random-Walker Algorithm

A simple approach is to reduce but not eliminate the rejection rate. The *random-walker* algorithm keeps track of boundary lattice sites and selects only these as target sites in the modified-Metropolis algorithm, eliminating the automatic rejection of non-boundary sites in the normal algorithm [3]. For large cells and short interaction ranges, speed gains are substantial. For small cells or subcells, the increase in memory required outweighs the speed gains.

4. Parallelization for Large Simulations

Current practical single-processor GGH simulations can handle about 10^5cells. However, a full model of the morphogenesis of a complete organ or an entire embryo requires the simulation of 10^6–10^8cells. We also constantly demand more lattice detail and complex cell behaviors in our simulations. This need for speed suggests using distributed computing, which is widely available.

Parallel implementations of the GGH are an active area of research. The main difficulty in GGH parallelization is that the effective energies are non-local and updating is stochastic. For example, when a given cell crosses between nodes, any modification to it requires parameters passing between nodes or computations will use stale parameters.[2] Because in most current distributed computers (unlike in an IBM SP-1 or SP-2), processing is fast and interprocessor communications slow, a naive parallelization will cause the processes to spend almost all their time waiting rather than calculating.

[2]Studying the effect of stale parameter on GGH simulations would be a very valuable research project.

4.1. Checkerboard Algorithm

The standard approach to parallelizing Ising-type models is to divide the lattice into equivalent subgrids (with some overlap) one per processor. Each processor subdivides its subgrid into a checkerboard of sublattices [22], defined so that an update in one sublattice affects only the nearest-neighbor sublattices. We then define a set of non-interacting sublattices, *e.g.*, every 9th square in 2D or every 27th in 3D. If each processor then visits the same sublattice set and makes one update in each, the updates are guaranteed to be independent. The processors can then determine updates affecting neighboring processors, accumulate them and pass them synchronously, change the sublattice set and repeat. For examples of these methods see [4]. Gains from parallelization increase with the size of the subgrid per processor (which reduces the message-passing overhead) and decrease with the interaction range (which increases the message-passing overhead). To date, no one has implemented a fully-optimized checkerboard version of the GGH model.

4.2. Parallel Rejection-Free Algorithms

Rejection-free methods can use basic checkerboard parallelization, in which all processors perform equivalent computations, or can use an asymmetrical master–slave configuration, in which all processors can see the same lattice or in which the lattice is partitioned into subgrids. The master node (or nodes) determines all allowable updates, and delegates calculating the rates to slave nodes, which pass the rates back to the master node for steps 4 through 7 in the algorithm in section 3.1. Again, checkerboarding greatly improves the efficiency of the calculations. As far as we know, these methods have never been implemented in the GGH context.

5. Model Sharing Support

Because of the great simplicity of the GGH model's core algorithms (the GGH Hamiltonian and the modified Metropolis algorithm), a basic program that runs the GGH model takes less than one hundred lines of Fortran or C++ code. A result of this simplicity is that most of the major users of the GGH model wrote their own proprietary software without separating the individual model they were implementing from their GGH modeling framework. Current GGH simulations increasingly focus on complex cell behaviors, requiring detailed descriptions of the cell-cycle, transcription, and regulatory, metabolic and signaling networks. As the GGH model has accreted new capabilities, these proprietary versions have become more complex, with multiple, incompatible versions even within groups, making the replication of published results substantially more difficult and the adoption of new GGH extensions considerably slower [10, 11, 12, 26, 14]. Our efficiency would be much greater if our GGH implementations could either incorporate new methodologies in a simple way, or could communicate with programs that implement them, saving us from having to reinvent each-others' work and allowing us to seamlessly combine the efforts of different researchers. As other, non-GGH agent-based cell-level models emerge, we would like to compare their results to those of the GGH model for validation purposes. In addition, increasing numbers of computationally-non-sophisticated scientists wish to develop GGH models. For these potential users, hard-coding complex initial conditions and interactions represents an insuperable barrier.

5.1. GGH Packages

Several groups have recently released open-source, extensible GGH modeling packages which allow less-sophisticated users to build simulations of a wide range of morphogenic phenomena. Wide adoption of a small number of these choices will enormously facilitate model development and sharing. We will briefly discuss the *Tissue Simulation Toolkit* [19], a modeling *library* which requires hard-coding of user modifications, and *CompuCell3D* [13], a modeling *environment*, which allows users to write models using high-level abstractions. Both allow users to reproduce published results and share new algorithms relatively painlessly, opening the field of GGH modeling to a much broader audience.

5.1.1. CompuCell3D. The CompuCell3D modeling environment (currently released under the SimTk license[3]) was developed collaboratively by groups at the University of Notre Dame[4] and the Biocomplexity Institute at Indiana University, Bloomington.[5] It supports the full GGH model, with unlimited numbers of chemical fields and complex diffusion equations. Instead of writing Fortran or C++ code, CompuCell3D users specify models using an XML-based markup language and run simulations from a flexible graphical *player*. CompuCell3D offers high-level Python scripting abilities that allow users full control of system-level variables and permit complex, conditional and time-dependent behavior specification without the need for hard-coding. CompuCell3D is under active development. Its development team welcomes participation by current and potential users in all aspects of its development. See also three simulations: MovII.4.2, MovII.4.3 and MovII.4.4 from the accompanying DVD.

5.1.2. Tissue Simulation Toolkit. Merks' open-source GGH library, the Tissue Simulation Toolkit (*TST*),[6] provides modules for simple 2D GGH simulations, with sample programs (available on the accompanying DVD) that allow users to reproduce published simulations. The TST includes a full range of GGH features and chemical fields. Visualization tools allow users to display the cell lattice by cell-type and chemical fields using color ramps and contour lines. The TST runs either interactively or in the background, making the package especially suitable for running large-scale parameter sweeps on computing clusters.

[3]https://simtk.org/home/compucell3d/.
[4]http://www.nd.edu/∼lcls/compucell/.
[5]http://biocomplexity.indiana.edu/.
[6]http://sourceforge.net/projects/tst.

5.2. Model Sharing

As other agent-based, cell-level models emerge, comparing results from different models will require a markup language that specifies cell-level models and cell behaviors in a generic way, independent of the simulation engine and modeling methodology. For example, to describe chemotaxis, the markup language might specify that epithelial cells of type C chemotax up TGF-β gradients with a sigmoidal response curve with saturation parameter α_1 and strength λ_1, and chemotax away from FGF-8 with a linear response with strength λ_2. Individual simulation packages would then implement this behavior in their own ways (perhaps, with some simplifications if they lacked specific features). CompuCell3D is already moving in this direction with its XML-based model specification. Furthermore, the XML-based markup language BioLogo (which is part of CompuCell3D) allows generic specification, coding and compiling of reaction-diffusion mechanisms. The Systems Biology Markup Language (*SBML*)[7] Level 3 is another markup language, which currently plans features for spatial, compartmentalized models.

Even more valuable than model sharing through markup languages would be model interoperability via *sockets* that allow communication between programs. Sockets would allow GGH implementations (such as the TST or CompuCell3D) to either incorporate their own engines to implement behaviors or communicate with other programs that do. Using sockets, a generic markup language could specify biological models in enough detail to allow sharing across model scales, connecting GGH-model implementations to microscopic models like BioSpice[8] and Systems Biology Workbench (*SBW*),[9] and to macroscopic models like Physiome.[10] The developers of CompuCell3D are currently working with the developers of SBW to implement sockets between their packages, allowing SBW to simulate the internal chemical pathways of the cell and CompuCell3D to simulate cells and intra-cellular behaviors. With a fully-defined markup language and set of sockets, a researcher building a CompuCell3D simulation would be able to use another researcher's existing model for internal cell behaviors, without recoding or reimplementation.

6. Conclusions

This chapter has presented some current areas of GGH model development and presented a view of its future, implicitly assuming that GGH modeling has a future. Fig.6 shows the number of papers that discuss or use GGH-related models. We would like to see an exponential rise rather than a peak followed by a leveling-off or a drop. The citation rate through 2005 increases roughly exponentially, albeit with a slow doubling time of five years, confirming a steady increase in interest in GGH models. As its power and flexibility increase through new methods, packages and

[7]http://sbml.org.
[8]http://biospice.org.
[9]http://sbw.sourceforge.net/.
[10]http://www.physiome.org/.

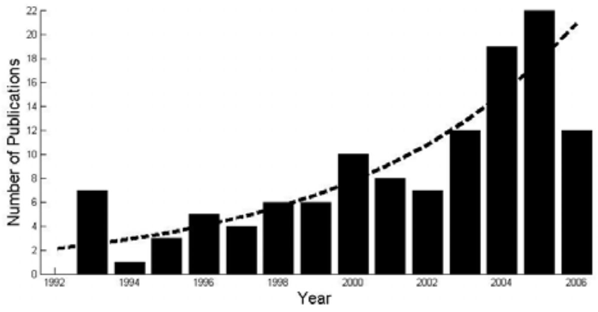

FIGURE 2. Number of publications using or citing GGH-related models by year (source: Web of Science, PubMed, ArXiv, *etc.*).

model sharing, we expect the GGH approach to become a standard method for cell-to-tissue level *in silico* biology, first replicating, then guiding *in vitro* experiments, and eventually leading to new experimental discoveries.

Acknowledgments

The leaf simulation project is a collaboration between R. M. H. Merks and G. T. S. Beemster, Y. Van de Peer and D. Inzé of the Department of Plant Systems Biology, VIB and Ghent University, Ghent, Belgium. R. M. H. Merks is supported by Marie Curie Intra-European Fellowship 2005-025084. A. Balter, N. J. Popławski, M. Swat and J. A. Glazier gratefully acknowledge support from the National Institutes of Health, National Institute of General Medical Sciences, grant 1R01 GM076692-01, an IBM Innovation Institute award, NASA Glenn Research Center, grant NAG 2-1619, the National Science Foundation, Division of Integrative Biology, grant IBN-0083653, and the Office of the Vice President for Research, the College of Arts and Sciences, the Pervasive Technologies Laboratories, the AVIDD and ODIN computer clusters and the Biocomplexity Institute at Indiana University.

References

[1] U. Börner, A. Deutsch, H. Reichenbach, and M. Bär. Rippling patterns in aggregates of myxobacteria arise from cell–cell collisions. *Phys. Rev. Lett.*, 89:078101, 2002.

[2] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz. A new algorithm for Monte Carlo simulation of Ising spin systems. *J. Comp. Phys.*, 17:10, 1975.

[3] F. P. Cercato, J. C. M. Mombach, and G. G. H. Cavalheiro. High performance simulations of the Cellular Potts model. In *20th International Symposium on High-Performance Computing in an Advanced Collaborative Environment*, page 28, 2006.

[4] N. Chen, J. A. Glazier, and M. S. Alber. A parallel implementation of the Cellular Potts model for simulation of cell-based morphogenesis. *Lect. Notes Comput. Sci.*, 4173:58, 2006.

[5] D. Dan, C. Mueller, K. Chen, and J. A. Glazier. Solving the advection–diffusion equations in biological contexts using the Cellular Potts model. *Phys. Rev. E*, 72: 041909, 2005.

[6] R. O. Erickson. Symplastic growth and symplasmic transport. *Plant Physiol.*, 82: 1153, 1986.

[7] K. A. Fichthorn and W. H. Weinberg. Theoretical foundations of dynamical Monte Carlo simulations. *J. Chem. Phys.*, 95:1090, 1991.

[8] M. C. Gibson, A. B. Patel, R. Nagpal, and N. Perrimon. The emergence of geometric order in proliferating metazoan epithelia. *Nature*, 442:1038, 2006.

[9] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution for coupled chemical reactions. *J. Comp. Phys.*, 22:403, 1976.

[10] P. Hogeweg. Evolving mechanisms of morphogenesis: on the interplay between differential adhesion and cell differentiation. *J. Theor. Biol.*, 203:317, 2000.

[11] P. Hogeweg. Computing an organism: on the interface between informatic and dynamic processes. *Biosystems*, 64:97, 2002.

[12] P. Hogeweg and N. Takeuchi. Multilevel selection in models of prebiotic evolution: compartments and spatial self-organization. *Orig. Life Evol. Biosph.*, 33:375, 2003.

[13] J. A. Izaguirre, R. Chaturvedi, C. Huang, T. Cickovski, J. Coffland, G. L. Thomas, G. Forgacs, M. S. Alber, H. G. E. Hentschel, S. A. Newman, and J. A. Glazier. Compucell, a multi-model framework for simulation of morphogenesis. *Bioinformatics*, 20: 1129, 2004.

[14] Y. Jiang, J. Pjesivac-Grbovic, C. Cantrell, and J. P. Freyer. A multiscale model for avascular tumor growth. *Biophys. J.*, 89:3884, 2005.

[15] K.-C. Lee. Rejection-free Monte Carlo technique. *J. Phys. A*, 28:4835, 1995.

[16] J. A. Lockhart. An analysis of irreversible plant elongation. *J. Theor. Biol.*, 8:264, 1965.

[17] H. Meinhardt. Morphogenesis of lines and nets. *Differentiation*, 6:117, 1976.

[18] R. M. H. Merks and J. A. Glazier. Dynamic mechanisms of blood vessel growth. *Nonlinearity*, 19, 2006.

[19] R. M. H. Merks and J. A. Glazier. A cell-centered approach to developmental biology. *Physica A*, 352:113, 2005.

[20] R. M. H. Merks, S. V. Brodsky, M. S. Goligorksy, S. A. Newman, and J. A. Glazier. Cell elongation is key to *in silico* replication of *in vitro* vasculogenesis and subsequent remodeling. *Dev. Biol.*, 289:44, 2006.

[21] T. Nagai and H. Honda. A dynamic cell model for the formation of epithelial tissues. *Philos. Mag.*, 81:699, 2001.

[22] M. E. J. Newman and G. T. Barkema. *Monte Carlo methods in statistical physics.* Oxford University Press, Oxford, 3rd edition, 1999.

[23] J. H. Priestley. Studies in the physiology of cambial activity. II. The concept of sliding growth. *New Physiol.*, 29:96, 1930.

[24] W.-J. Rappel, A. Nicol, A. Sarkissian, H. Levine, and W. F. Loomis. Self-organized vortex state in two-dimensional *Dictyostelium* dynamics. *Phys. Rev. Lett.*, 83:1247, 1999.

[25] T. Rudge and J. Haseloff. A computational model of cellular morphogenesis in plants. *Lect. Notes Comput. Sci.*, 3630:78, 2005.

[26] N. J. Savill and J. A. Sherratt. Control of epidermal stem cell clusters by Notch-mediated lateral induction. *Dev. Biol.*, 258:141, 2003.

[27] J. Starruß, T. Bley, and A. Deutsch. A new mechanism for swarming pattern formation of *Myxococcus xanthus*. Preprint, 2007.

[28] M. Zajac, G. L. Jones, and J. A. Glazier. Model of convergent extension in animal morphogenesis. *Phys. Rev. Lett.*, 85:2022, 2000.

Ariel Balter
Biocomplexity Institute and Department of Physics, Indiana University,
Swain Hall West 117, 727 East Third Street, Bloomington, Indiana 47405-7105, USA
e-mail: `abalter@indiana.edu`

Roeland M. H. Merks
Department of Plant Systems Biology,
VIB Technologiepark 927, B-9052 Ghent, Belgium, and
Department of Molecular Genetics, Ghent University, Ghent, Belgium
e-mail: `roeland.merks@psb.ugent.be` or `post@roelandmerks.nl`

Nikodem J. Popławski
Biocomplexity Institute and Department of Physics, Indiana University,
Swain Hall West 117, 727 East Third Street, Bloomington, Indiana 47405-7105, USA
e-mail: `nipoplaw@indiana.edu`

Maciej Swat
Biocomplexity Institute and Department of Physics, Indiana University,
Swain Hall West 117, 727 East Third Street, Bloomington, Indiana 47405-7105, USA
e-mail: `mswat@indiana.edu`

James A. Glazier
Biocomplexity Institute and Department of Physics, Indiana University,
Swain Hall West 159, 727 East Third Street, Bloomington, Indiana 47405-7105, USA
e-mail: `glazier@indiana.edu`