Bioinformatics, 31(14), 2015, 2340–2347
doi: 10.1093/bioinformatics/btv147
Advance Access Publication Date: 13 March 2015
Original Paper

OXFORD

Systems biology

LBIBCell: a cell-based simulation environment for morphogenetic problems

Simon Tanaka[1,2,*], David Sichau[1] and Dagmar Iber[1,2,*]

[1]Department for Biosystems Science and Engineering, ETH Zurich, Mattenstrasse 26, 4058 Basel, Switzerland and [2]Swiss Institute of Bioinformatics, Basel, Switzerland

*To whom correspondence should be addressed.

Associate Editor: Robert Murphy

Received on June 13, 2014; revised on March 2, 2015; accepted on March 10, 2015

Abstract

Motivation: The simulation of morphogenetic problems requires the simultaneous and coupled simulation of signalling and tissue dynamics. A cellular resolution of the tissue domain is important to adequately describe the impact of cell-based events, such as cell division, cell–cell interactions and spatially restricted signalling events. A tightly coupled cell-based mechano-regulatory simulation tool is therefore required.

Results: We developed an open-source software framework for morphogenetic problems. The environment offers core functionalities for the tissue and signalling models. In addition, the software offers great flexibility to add custom extensions and biologically motivated processes. Cells are represented as highly resolved, massless elastic polygons; the viscous properties of the tissue are modelled by a Newtonian fluid. The Immersed Boundary method is used to model the interaction between the viscous and elastic properties of the cells, thus extending on the IBCell model. The fluid and signalling processes are solved using the Lattice Boltzmann method. As application examples we simulate signalling-dependent tissue dynamics.

Availability and implementation: The documentation and source code are available on http://tanakas.bitbucket.org/lbibcell/index.html

Contact: simon.tanaka@bsse.ethz.ch or dagmar.iber@bsse.ethz.ch

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

During morphogenesis, tissue grows and self-organizes into complex functional units such as organs. The process is tightly controlled, both by signalling and by mechanical interactions. Long-range signalling interactions in the tissues can be mediated by diffusible substances, called morphogens, and by long-range cell processes (Restrepo *et al.*, 2014). The dynamics of the diffusible factors can typically be well described by systems of continuous reaction-advection-diffusion partial differential equations (PDEs). The appropriate tissue representation depends on the relevant time scale. For a homogeneous isotropic embryonic tissue, experiments show that the tissue is well approximated by a viscous fluid on long time scales (equilibration after 30 min to several hours) and by an elastic material on short time scales (seconds to minutes) (Forgacs *et al.*, 1998). However, biological control typically happens on a shorter time scale, and many cellular processes such as cell migration and adhesion, cell polarity, directed division, monolayer structures and differentiation cannot be cast into a continuous formulation in a straight-forward way. A number of cell-based simulation techniques at different scales and different level of detail have been developed to study these processes; here, we discuss main representatives for each category.

The *Cellular Potts model*, introduced by Graner and Glazier (1992), is solved on a lattice, with each lattice point holding a generalized spin value denoting cell identity. Similar to the Ising model, Hamiltonian energy functions are formulated and minimized using a Metropolis algorithm. It has been applied to a multitude of problems and is implemented in the software *CompuCell3D* (Swat *et al.*, 2012).

However, the correspondence between the biological problem and the Hamiltonian, the temperature and the time step is not always straightforward.

The *subcellular element model* divides cells into subcellular elements, which are represented by computational particles. The elements interact via interacting potentials which are subject to modelling. The motion of the elements is governed by overdamped Langevin dynamics, such that the method is mesh-free. The framework was first introduced by Newman (2005) and later applied by Sandersius *et al.* (2011a, b). This approach allows for detailed biophysical modelling, both in 2D and 3D.

The *spheroid model* developed by Drasdo *et al.* (2007) assumes that cells in unstructured cell populations are similar to colloidal particles. The cells are modelled as point particles, hosting interaction potentials. Their motions consist of a random and a directed movement. Neighbouring cells form adhesive bonds, which are represented using models borrowed from contact mechanics, such as e.g. the Johnson–Kendall–Roberts model (Chu *et al.*, 2005). Many cellular processes such as cell shape change, division, death, lysis, cell–cell interaction and migration have been successfully translated into the spheroid model (Drasdo *et al.*, 2007). Intra- and extracellular diffusion has not yet been introduced and implemented. The spheroid model extends efficiently to 3D, and it has been implemented in the open-source framework *CellSys* (Hoehme and Drasdo, 2010).

The *vertex model* uses polygons (or polyhedra in 3D) to represent cells in densely packed tissues, e.g. in *Drosophila* wing disc epithelia (Farhadifar *et al.*, 2007). For each vertex, forces are computed—either via a potential or directly. The vertices are moved subsequently according to overdamped equations of motion or via a Monte Carlo algorithm. The model is implemented in the open-source software *Chaste* (Pitt-Francis *et al.*, 2009).

The *viscoelastic cell model* (also called IBCell models) presented in Rejniak *et al.* (2004) and Rejniak (2007) uses the immersed boundary (IB) method (Peskin, 2003) to represent individually deformable cells as immersed elastic bodies. The cytoplasm and the extracellular matrix and fluid are represented by a viscous incompressible fluid. In this framework, a vast amount of biological processes such as cell growth, cell division, apoptosis and polarization has been realized. The model was applied to study tumour and epithelial dynamics. Due to the high level of detail, the viscoelastic cell model is computationally expensive and has not yet been implemented in 3D.

The software framework *VirtualLeaf* with explicit cell resolution, available in 2D, has been introduced in Merks *et al.* (2011). Although the cell representation is similar to vertex cell models, the dynamics is realized by minimizing a Hamiltonian using a Monte Carlo algorithm. The model assumes rigid cell walls, which is appropriate for plant morphogenesis.

For many morphogenetic phenomena, which arise from a tight interaction between the biomolecular signalling and the tissue physics, an explicit computational representation of the cell shapes is required. Here, we present a flexible software framework based on the IBCell model, which, as a novelty, permits to tightly couple biomolecular signalling models to a cell-resolved, physical tissue model. The core components and the general approach of the model are described in the second section. In the third section, the software and the main functionalities are described in detail. Application examples are given in the fourth chapter to demonstrate the framework's capabilities.

2 Approach

Our approach permits the coupled simulation of tissue and signalling dynamics. To describe the tissue dynamics, the viscoelastic cell model needs to represent both the cellular structures and their elastic properties, as well as the viscous behaviour of the cytoplasm and of the extracellular space surrounding the cells. The model therefore rests on three core parts: the representation of cells, the representation of the fluid and the fluid-structure interaction, and the coupling of the tissue part to the signalling model. To describe the interaction between the viscous fluid and the elastic structures, which are immersed in the fluid, we use Immersed Boundary method (Peskin, 2003) as previously implemented in the viscoelastic cell model, also called IBcell model (Rejniak *et al.*, 2004; Rejniak, 2007). To solve the viscous fluid behaviour, we use the Lattice Boltzmann method, which is an efficient mesoscopic numerical scheme, originally developed to solve fluid dynamics problems (Chen and Doolen, 1998). The method has previously been successfully applied to reaction-diffusion equations, such as Turing systems (Ponce Dawson *et al.*, 1993), as well as to coupled scalar fields such as temperature (Guo *et al.*, 2002). The method was for the first time combined with the Immersed Boundary method (Peskin, 2003) by Feng and Michaelides (2004), and has later been used to study red blood cells in flow by Zhang *et al.* (2007). In the following, we provide an overview of the implemented methods; the implementation details are given in Section 3.

2.1 Cell representation

Cells are represented as massless, purely elastic structures, which are described by sets of geometry points forming polygons. The geometry points are connected via forces. In a first approximation, the elastic structures can be identified to represent the elastic cell membranes. However, more elastic structures can be added to the intra- and extracellular volume to mimic the viscoelastic properties of the cytoskeleton or the extracellular matrix. The user can implement biological mechanisms which operate on the cell representations. For example, a new junction to a neighbouring cell might be created when the distance between two neighbouring cell boundaries falls below a threshold distance. Similarly, a junction might be removed when overly stretched.

2.2 Fluid and fluid-structure interaction

The viscoelastic cell model represents the content of cells (the cytoplasm) as well as the extracellular space (the interstitial fluid and the extracellular matrix) as a viscous, Newtonian fluid. The intra- and extracellular fluids interact with the elastic membrane, i.e. the fluids exert force on the membrane, and the membrane exerts force on the fluids. Furthermore, the velocity field of the fluid, which is induced by the forces, moves and deforms the elastic structures. This interaction, well-known as fluid-structure interaction, lies at the heart of the tissue model. Forces (e.g. membrane tension or cell–cell forces) acting on these points are exerted on the fluid by distributing the force to the surrounding fluid. Due to the local forcing, the fluid moves. At this step, the membrane point is advected passively by the fluid. As a result the forces need to be re-evaluated on the points. By repeating the forcing-advection steps, the interaction is realized iteratively.

As a result of this iterative process, the (elastic) structures are coupled to the (viscous) fluid. Depending on the parameterization, this model allows to describe either elastic, or viscous, or viscoelastic material behaviour. The upper part of Figure 1 illustrates the Immersed Boundary interaction. The implemented Immersed Boundary kernel function has bounded support, i.e. each geometry point influences and is influenced only its immediate neighbourhood. Here, the dimension of the kernel function is four by four (cf. Fig. 1). The fluid equations are solved using the Lattice Boltzmann method (Chen and Doolen, 1998), which is described in detail in the

Supplementary material (Section 6). The Reynolds number is typically $\ll 1$; hence, the regime is described by Stokes flow (The Reynolds number reads $Re = \frac{UL}{\nu}$, with U being a characteristic velocity, L a characteristic length scale and ν the kinematic viscosity. Assuming $L = 10^{-3}\,[m]$, $U = 10^{-8}\,[m/s]$ and $\nu = 10^{1} \ldots 10^{2}\,[m^2/s]$, then $Re = 10^{-13} \ldots 10^{-12}$ can be estimated (Forgacs *et al.*, 1998)).

2.3 Signalling

The signalling network is represented as a system of reaction-advection-diffusion processes. The elastic membranes may act as no-flux boundaries for compounds which only exist in the extra- or intracellular volume, respectively. The reaction-advection-diffusion solvers can be equipped with potentially coupled reaction terms in order to model signalling interactions of diffusing factors. Depending on the model, the signalling may impact the tissue dynamics. This can be done, for instance, by making the mass source of the fluid dependent on the values of the reaction-advection-diffusion solvers such that the tissue expands locally (cf. Fig. 1). Furthermore, the diffusing compounds can be individually configured to diffuse freely across the entire domain, or only inside or outside the cells (e.g. using no-flux boundary conditions for the cell membranes).

3 Software

3.1 Cell representation

The cell geometries consist of two elements, the `GeometryNodes`, which act as the IB points, and the `Connections`, connecting pairs of `GeometryNodes`. A simplified cell is visualized in Figure 2. The `Connections` are attributed with a `domainID` flag, which is an identifier for the surrounded domain (respecting the counter-clockwise directionality convention). The domain identifier on the other side (on the right hand side) is zero by convention, representing the interstitial space. The `domainID` of the connections are copied to the fluid and reaction-advection-diffusion solvers. Moreover, the `domainID's` are associated with a cell type flag, `cellType`. By applying custom differentiation rules, the `cellType` of individual cells may be changed according to custom criteria; otherwise the all cells default to `cellType = 1` (with `cellType = 0` being the interstitial space, again). In this way, the reaction terms and the mass sources may be made dependent on specific cells or specific cell types.

3.2 User-provided solvers

The user can add the following routines: `MassSolverXX`, `CDESolverXX` and `BioSolverXX` (`XX` being a name to be chosen). The `MassSolverXX`—as described earlier—adds or subtracts mass from/to the fluid solver. The `CDESolverXX` is used to implement the reaction terms of the signalling models. Finally, the `BioSolverXX` can be used to execute biologically motivated operations on the geometry and the forces. Such an operation might be cell division, which is discussed in more detail in Section 3.5.4. Figure 3A summarizes the most important classes and their interactions. The classes which are subject to customization are shaded. In order to add a new customized routine (e.g. a mass modifying solver `MassSolverXX`, a reaction-advection-diffusion solver `CDESolverXX` or a biologically motivated solver `BioSolverXX`),

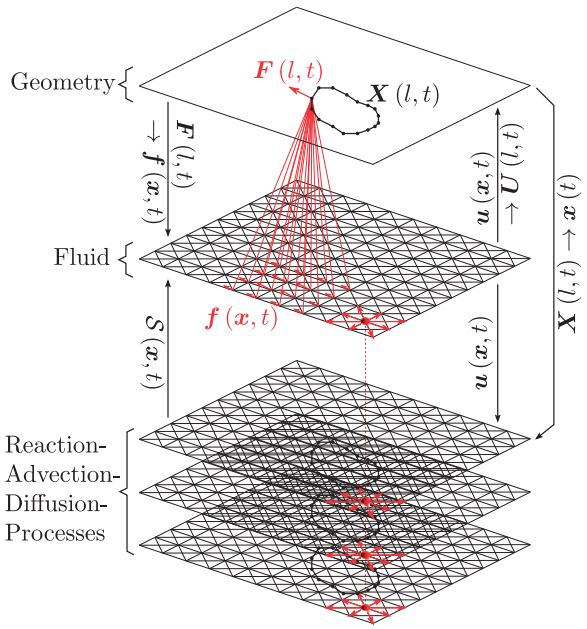

Fig. 1. Algorithm overview. The algorithm consists of three coupled layers. The geometry $X(l,t)$ (top part, discussed in more detail in Fig. 2) is used to compute the forces $F(l,t)$ acting on each of the geometry nodes. These forces, which do not necessarily coincide with a lattice point, are scattered to the fluid lattice (middle part) using the IB method kernel function, $F(l,t) \rightarrow f(t)$. After advancing the fluid solver by one time step, the velocity is interpolated to the geometry node position using the same kernel function, $u(x,t) \rightarrow U(l,t)$. The geometry nodes are moved according to their velocity $U(l,t)$, and the iteration is restarted. The velocity $u(x,t)$ of the fluid lattice is also copied to the reaction-advection-diffusion solvers (PDE), together with the position $X(l,t) \rightarrow x(t)$ of the geometry. The state of the reaction-advection-diffusion solvers, which are used to model signalling, may be used to compute mass sources $S(x,t)$ for the fluid solver

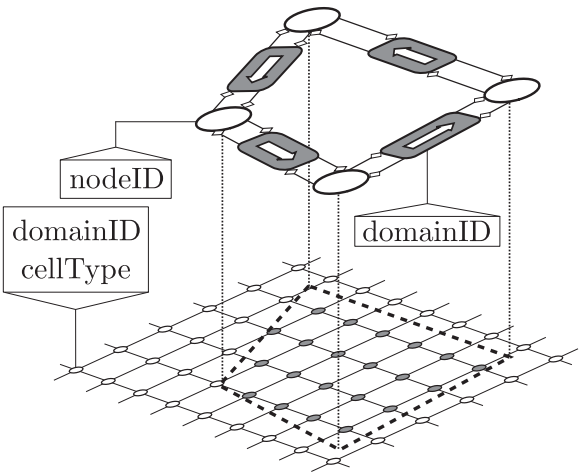

Fig. 2. Elements of the geometry representation. The cells are closed polygons, consisting of geometry nodes (discussed in the top part) and connections (shaded boxes in the top part) between each two geometry nodes. Each connection stores two references to its preceding and successive geometry nodes, and *vice versa* each geometry node stores two references to its preceding and successive connection (visualized by aggregation arrows in the top part). Directionality of the polygon is counter-clockwise by convention. Each geometry node has a unique, immutable nodeID attribute, which is allocated internally upon creation of a new geometry node. Each connection features a domainID attribute, which denotes the domain identifier of the domain on the left hand side. The domain identifier on the right hand side is by definition zero, representing the extracellular space. Using the domainID of the connections, the domainID of the lattice nodes is automatically set (lower part). Additionally, each domainID is associated with a cellType. The behaviour of the MassSolverXX, BioSolverXX and CDESolverXX can be made dependent on the domainID and/or cellType attributes by the user

the user needs to inherit from their respective virtual base classes (cf. Fig. 3B). Figure 4 visualizes the routines, which are called iteratively by the `SimulationRunner` (cf. Fig. 3A).

3.3 Input and output

The communication to the user is achieved via the loading and dumping of configuration files. A general configuration file contains the global simulation parameters, such as the simulation time, the domain size, the fluid viscosity and the diffusion coefficients for the reaction-advection-diffusion solvers. The geometry points and the corresponding geometrical connections are stored in a geometry file. A third file contains the forces, including forces between a pair of geometry points, freely defined forces or spatially anchored points. The fluid and reaction-advection-diffusion solver states may be written either to .txt files or in .vtk format and can be post-processed with third-party software (e.g. Matlab or ParaView). Optionally, the solver states can be saved in a loadable format to resume the simulation.

3.4 Physical processes

3.4.1 Viscous and elastic behaviour

The viscous behaviour is implemented using a representation of an incompressible fluid (solved using the Lattice Boltzmann method, cf. Supplementary material), which converges to the Navier–Stokes equation in the hydrodynamic limit. The fluid is solved on a regular Cartesian and Eulerian grid. The membranes are represented by sets of points, which are connected to form closed polygons. A variety of forces may act on the membrane nodes, such as e.g. membrane tensions (cf. Section 3.4.3). The interaction between the fluid and the elastic structures is formulated using the Immersed Boundary method (cf. Supplementary material). The membrane points move according to the local fluid velocity field in a Lagrangian manner.

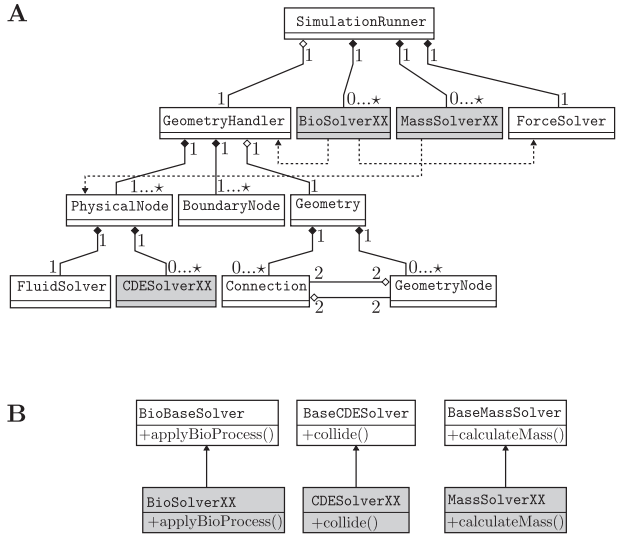

Fig. 3. Simplified UML diagram of important classes. The classes which have to be provided by the user are shaded. XX refers to an arbitrary solver name. **(A)** The `SimulationRunner` controls the execution of the simulation. The `GeometryHandler` has a collection of `PhysicalNodes`, representing the lattice, a collection of `BoundaryNodes` which are woven into the lattice, and a `Geometry` object. The latter contains the cell's geometric information, namely the `GeometryNodes` and the `Connections`. The `GeometryNodes` and the `Connections` each have two references of the preceding and successive elements, as also explained in Figure 2. `BioSolverXX` obtains references from the `GeometryHandler` and the ForceSolver to alter states accordingly. Similarly, the `MassSolverXX` obtains a reference to the lattice and adds mass sources to the fluid. **(B)** To implement new custom routines, the user must inherit from provided base classes (from `BioBaseSolver` for biologically motivated routines, from `BaseCDESolver` for reaction-advection-diffusion processes, and from `BaseMassSolver` for mass modifying routines)

3.4.2 Reaction-diffusion of biochemical compounds

The biochemical signalling can be described by sets of coupled reaction-diffusion PDEs. Similar to the fluid equations, these equations are solved on a regular Cartesian and Eulerian grid (solved using the Lattice Boltzmann method, cf. Supplementary material). The concentrations of the compounds can be accessed by other solvers, for example to make other processes such as cell division dependent on signalling factors. The cell boundaries can be chosen to be either invisible to the diffusing compounds or to be no-flux boundaries. To account for advection, the fluid velocity field is directly transferred from the fluid solver since the fluid and the reaction-diffusion lattices coincide spatially. The coupling of the solvers is visualized in Figure 1.

3.4.3 Forces

Forces are an integral part of the simulation environment. A force is always connected to a membrane point. Any type of conservative force (which can be derived from a potential) can easily be implemented. Currently, the following types of forces are implemented:

- spring force between two geometrical nodes
- spring force between a geometrical node and a spatial anchor point
- free force acting on a geometrical node
- horizontally or vertically sliding force (thus enforcing only the y or x coordinate, respectively)
- constant force between two geometrical nodes

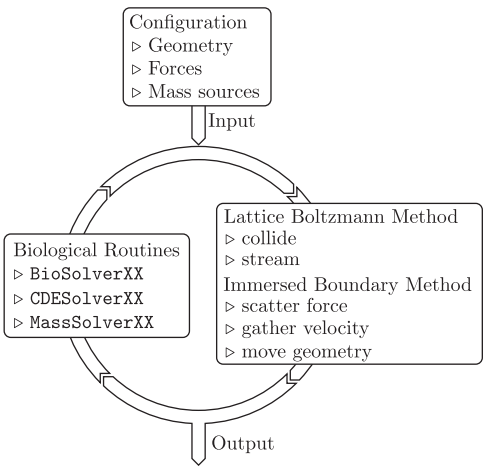

Fig. 4. Iterative processing in the solver. At initiation, the library loads the user-provided configuration files (containing global simulation parameters, initial geometry, initial forces). During each iteration, the library's class `SimulationRunner` (cf. Fig. 3A) successively calls the physical routines (the Lattice Boltzmann method to solve the fluid and reaction-advection-diffusion processes, and the Immersed Boundary method to solve the fluid-structure interaction) and the biological routines (biologically motivated re-arrangement of the geometry, modifications of the forces, etc.). The current configuration and optionally the entire solver states can be saved at a chosen frequency

Application examples include constant forces between two geometrical nodes that can be used to model constant membrane tension, which leads to the minimization of a cells perimeter (discussed in Section 3.5.2). Moreover, a geometrical point can dynamically explore its local neighbourhood and establish a force to another geometrical point *from another cell*, thus, mimicking cell–cell junctions (discussed in Section 3.5.3).

3.5 Biological processes

The biological solvers (BioSolver) accommodate the functionalities that are related to biological processes. These processes may be mostly related to modifications of the forces and the geometry. The BioSolver has full access to the compound concentrations. Furthermore, it is aware of the cells, whose geometries are stored individually. This enables the BioSolver to compute cell areas and averaged or integrated compound concentrations. Because all cells are individually tagged, cell behaviour can be made dependent on cell identity. Additionally to the cell identity, cells also carry a cell type tag, which can be changed depending on run-time conditions. This latter functionality can be used to model cell differentiation.

Consider a cell division event as an example. Here, a division plane has to be chosen. The choice of its position and direction is subject to the user's model: the cell division plane might be set perpendicular to the cell's axis of strongest elongation. Next, the cell has to be divided, which requires the removal of the corresponding geometrical connections, and the insertion of new geometrical nodes and connections to close the divided cells.

Note that the concentration fields of the compounds, as well as the velocity- and pressure fields of the fluid solver are not directly altered in the biological module.

3.5.1 Control of cell area

Depending on the biological model of the user, the cell area has to be controlled. By assuming that a cell might change its spatial extent in the third dimension, the area might shrink or expand as a response to forces exerted by its neighbouring cells, which can effectively be modelled as an 'area elasticity'. In the limiting case, the cell resists external forces, maintains its area and only reacts with changes of the hydrostatic pressure. In general, to control the area of cells, the reference area for each cell needs to be adapted. The reference area acts as a set point for a simple proportional controller, i.e. the local mass source $\mathcal{S}_k$ in the cell k is proportional to the area difference between the current cell area $A_k(t)$ and the set point area A_k^0:

$$\mathcal{S}_k = \alpha\left(A_k^0 - A_k(t)\right) \tag{1}$$

where α is a proportional constant. More advanced control methods, such as e.g. proportional-integral control methods, can be realized easily.

This approach of controlling the cell area can also be used to let cells grow or shrink in a controlled way, i.e. a cell differentiating into a hypertrophic cell type may grow in volume. Implementing this process would be as simple as setting the new target area as set point area. The area controller will bring the cell close to its new area.

3.5.2 Membrane tension

The definition of forces acting between pairs of membrane points allows for simulating the cell's membrane tension. By default, a constant contracting force $\boldsymbol{F_i}$ with magnitude φ^m is applied to every pair of neighbouring membrane points. Hence, the resulting force on membrane point i is composed of a force pointing to its preceding membrane point $i-1$, and a force pointing to its successive membrane point $i+1$:

$$\boldsymbol{F_i^m} = \varphi^m\left(\frac{\boldsymbol{x}_{i-1} - \boldsymbol{x}_i}{|\boldsymbol{x}_{i-1} - \boldsymbol{x}_i|} + \frac{\boldsymbol{x}_{i+1} - \boldsymbol{x}_i}{|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i|}\right) \tag{2}$$

This approach can be interpreted as an actively remodelled membrane: when stretched, new membrane is synthesized in order to not increase the membrane tension on longer time scales (hours). On the other hand, excessive membrane is degraded to abide the membrane tension. Therefore, the membrane tension minimizes the cell's perimeter. Because the intracellular fluid (and thus the cell area) is conserved in the absence of neighbouring cells and active mechanisms (c.f. Section 3.5.1), the cell assumes a circular shape. On short time scales (seconds), the passive (non-remodelled) elastic membranes can be modelled by using Hookean spring potentials. The membrane tension will then be proportional to deviation from the resting membrane perimeter. In both cases, the membrane is flexible (i.e. has no bending stiffness); if bending stiffness should be required by the user, this can be easily realized in a custom BioSolver.

The implementation of membrane tension needs to consider the geometry remeshing. Whenever a new membrane point is inserted, it needs to get connected to its neighbours instantly, because the cell will be overly stretched in the absence of membrane tensions. A membrane point's forces need to be removed upon its removal. Algorithmically, this is realized by removing and reconstructing all membrane forces at every time step. At this point, the magnitude of the membrane tension can be made dependent on signalling factors.

`BioSolverMembraneTension` is an example of a class managing the membrane tensions with immediate remodeling, and `BioSolverHookeanMembraneTension` implements simple Hookean springs.

3.5.3 Cell junctions

A cell can create cell junctions to neighbouring cells. In the simplest case, each membrane point i uses the function `getGeometryNodesWithinRadiusWithAvoidanceClosest` to get the closest membrane point j of another cell, which is within a predefined cut-off radius $l^{\max}$, or zero if there is no such membrane point. Once a candidate membrane point fulfils the criteria, a new Hookean force $\boldsymbol{F_i}$ with a spring constant k^j and resting length l_0 is created:

$$\boldsymbol{F_i^{cj}} = \begin{cases} k^j \dfrac{\boldsymbol{x_j} - \boldsymbol{x_i}}{|\boldsymbol{x_j} - \boldsymbol{x_i}|}\left(|\boldsymbol{x_j} - \boldsymbol{x_i}| - l_0\right) & \text{if } |\boldsymbol{x_j} - \boldsymbol{x_i}| < l^{\max} \\ 0 & \text{else} \end{cases} \tag{3}$$

The cell junction forces are regularly (potentially not at every time step) deleted and renewed, where the frequency of cell junction renewal might reflect the cell junction synthesis rate.

The function `getGeometryNodesWithinRadiusWithAvoidance` returns all membrane points of another cell, which are within a predefined cut-off radius; the returned list might be empty. This opens up the possibility to introduce randomness by choosing the membrane point randomly from the candidate list. The probability to create a junction might depend on the junction length: the shorter, the higher the probability to form a new junction. Also the removal of membrane points might be randomized, and the probability made dependent on the junction length, i.e. overly stretched junctions are removed with higher probability. Even the membrane point whose junctions shall be updated might be chosen randomly. Again, the number of updated membrane nodes per time reflects the cell's limited cell junction synthesis activity.

The membrane points are internally stored in a fast neighbour list data structure, which is well suited for spatial range queries. `BioSolverCellJunction` is an example of a class responsible for cell junctions.

3.5.4 Cell division

The cell division functionality requires several steps. First, criteria will have to be defined which cells shall be divided. Criteria might be maximal cell area, maximal spatial expansion or biochemical signals. Once a cell committed for division, the cell division plane will have to be chosen. Again, how to chose the plane is subject to biological modelling. A frequently used rule is to use a plane defined by a random direction vector and the center of mass of the cell. However, different rules can be readily implemented, such as random directions drawn from non-uniform probability distributions (which, in turn, can be controlled, e.g. by signalling factor gradients) or division planes perpendicular to the longest axis (Minc *et al.*, 2011). In a next step, the two membrane segments are determined which intersect with the division plane; this is implemented in `getTwoConnections RandomDirection` or `getTwoConnectionsLongestAxis`. These two membrane segments are subsequently removed, and two new membrane segments across the cell are introduced, leading to a cut through the mother cell. Finally, a new domain identity number has to be given to one of the daughter cells; the other daughter cell inherits the domain identity number from the mother cell. The new domain identity number is set to the largest domain identity number plus one, and it is automatically copied to the physical grid. Both daughter cells by default inherit the cell type flag from the mother cell, which is also automatically copied to the physical grid.

The basic cell division functionality is implemented in the class `BioSolverCellDivision`.

3.5.5 Differentiation

Differentiation changes the cell type flag of the cells according to user-defined, biologically motivated rules. These rules might be based on the cell area, or on a signalling factor concentration, possibly integrated over the cell area. Once being committed for differentiation, the cell changes its cell type flag according to the rule. The new cell type flag will be automatically copied to the physical grid. The cell type flag can be used to make signalling dynamics, but also other biologically motivated processes dependent on the cell type.

The association between the domain identifier flags and the cell type flags is stored in the `cellTypeTrackerMap_`, which is a member of the `GeometryHandler`. This makes sure that all `BioSolverXX` classes have easy access to this information. A basic implementation of the differentiation control can be found in `BioSolverDifferentiation`.

3.6 Accuracy and performance

The Lattice Boltzmann schemes are second order accurate, and the explicit Immersed Boundary method is first order accurate in space and time. The internal data structure uses a fast neighbour list (cell list) implementation to optimize for range queries (e.g. searching for other cells in the local neighbourhood), which exhibits a search complexity of $\mathcal{O}(N)$, with N being the number of membrane points to represent the cells. Many iterative computations (Lattice Boltzmann and Immersed Boundary routines such as particle streaming and collision, gathering of velocity and scattering of force) are parallelized using the shared memory paradigm. However, a few computational steps cannot be parallelized. This is typically the case when write-operations occur on shared data structures, such as the data structures storing the geometry nodes and the force structs (e.g. in `ForceSolver::delete ForceType()` and `GeometryHandler ::computeAreas()`). Moreover, the geometry remeshing (refining and coarsening) functions as well as the data I/O are not parallelized, but are assumed to occur much less frequently than the actual fluid and reaction-advection-diffusion solvers. Therefore, since the fraction of sequential code is not negligible, the software should best be run on fast multi-core processors.

3.7 Tools, dependencies and documentation

A compiler with C++0x support (such as GCC 4.7 or higher) is required. The software depends on Boost (http://www.boost.org; 1.54.0 or higher), OpenMP, CMake (http://www.cmake.org) and vtk (http://www.vtk.org/; 5.8 or higher). The source code is extensively documented using Doxygen (http://www.stack.nl/dimitri/doxygen). Git (http://git-scm.com) is used for version control. The software has only been tested on linux operating systems.

3.8 Availability

The documentation and source code are available on http://tanakas.bitbucket.org/lbibcell/index.html.

4 Application examples

4.1 Cell division, differentiation and signalling

To demonstrate the capabilities of the software, we first consider a tissue model with cell-type specific cell division and signalling-dependent differentiation (Fig. 5). In the beginning, a circular cell with radius $R = 10$ is placed in the middle of a quadratic 400 by 400 domain (Fig. 5A). Iso-pressure boundary conditions are set at the border of the domain. The initial cell is of red cell type, which is proliferating at a high rate. When considering a single layer epithelium, mass uptake, which is needed for modelling cell growth and finally proliferation, is assumed to occur from the apical cavity through the apical membrane. Additionally, the initial cell secretes a signalling factor $\mathcal{I}$ which inhibits differentiation of the red cell type into the green cell type. Once the cell area doubled, the cell is divided in a random direction (cf. Fig. 5B). The daughter cells inherit the cell type, but only the mother cell continues to express the signalling molecule $\mathcal{I}$. All cells of red type integrate the concentration of $\mathcal{I}$ over their area. For low signalling levels, the red cell type differentiates into the green cell type. The green cell type does not grow and only divides if external forces stretch the cell. In Figure 5C, the daughter cell's signalling level dropped after cell division, and differentiation occurred. After several rounds of cell division, a tissue starts to form (cf. Fig. 5D). The cells close to the secreting initial cell remain protected from differentiation, whereas more distant cells differentiate irreversibly. Due to the randomly chosen cell division axis, it might happen that the proliferating red cells get trapped (cf. Fig. 5E). The expression of $\mathcal{I}$ is switched off at time $t = 5000$, thus leading to complete differentiation shortly after (cf. Fig. 5F). After proliferation stopped, the cells slowly rearrange because cell–cell junctions are broken if overly stretched, and new junctions are formed [according to Equation (3)]. At the boundary of the tissue, the cells try to reach a spherical shape, while in the middle mainly characteristic penta- and hexagonal shapes emerge (cf. Fig. 5F and Supplementary file S6.4).

4.2 Turing patterning on growing cellular domains

To demonstrate the importance to investigate morphogenic signalling hypotheses on dynamically growing domains with cellular resolution, we solved a reaction-diffusion system, featuring the well-known diffusion-driven Turing instability (Turing, 1952), on a proliferating tissue.

Figure 6A illustrates the interaction between a ligand L and its receptor R. Here, we assume that one ligand dimer molecule L binds to two receptors R, forming the complex R^2L which induces upregulation of the receptor on the membrane (e.g. Bellusci *et al.*, 1997). Unbound receptor is turned over at a linear rate. The ligand can diffuse freely across the tissue and the entire domain, whereas the diffusion of the receptor is limited to a single cell's apical surface and is much slower. The dynamics can be formulated as a system of non-dimensional PDEs:

$$\partial_t R = \Delta R + \gamma(a - R + R^2 L) \tag{4}$$

$$\partial_t L = d\Delta L + \gamma(b - R^2 L) \tag{5}$$

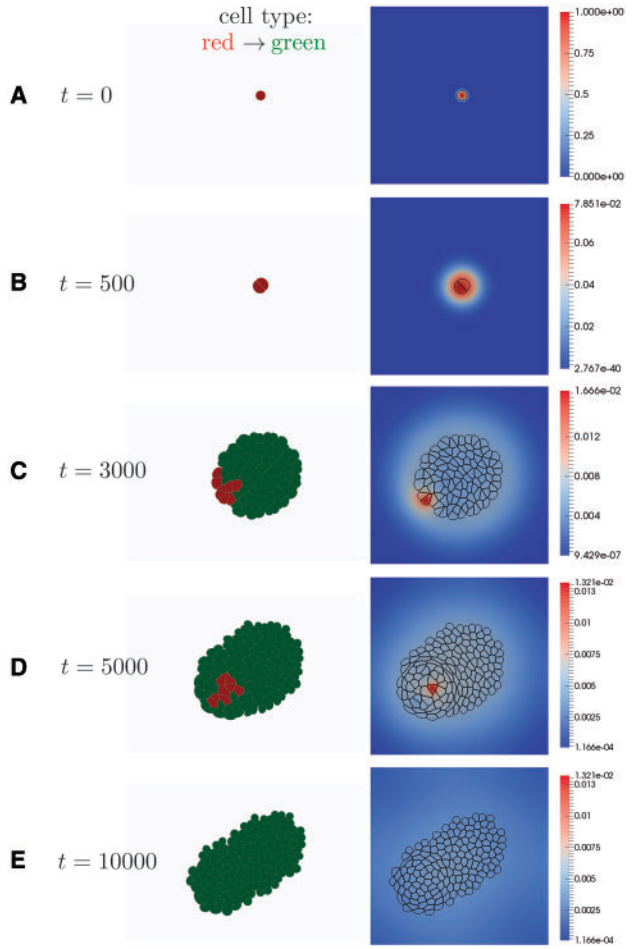

Fig. 5. Cell division, differentiation and signalling. (**A**) The initial configuration consists of a single, circular cell of type red. The red cell type proliferates at a high rate. The initial cell is tagged and expresses a signalling molecule $\mathcal{I}$ which inhibits differentiation. (**B**) The first cell division occurs. The division axis is chosen randomly. The daughter cell inherits the cell type from the mother cell, but only the mother cell keeps expressing the signalling molecule $\mathcal{I}$. (**C**) The signalling level (the spatially integrated concentration of the signalling molecule) drops in cells far away from the initial cell and differentiation into the green cell type occurs. The green cell type does not grow intrinsically, and only divides if overly stretched by external forces. (**D**) The highly proliferating red cells are trapped in the forming tissue due to the randomly chosen cell division axis. At $t = 5000$, the expression of the differentiation inhibiting molecule $\mathcal{I}$ is switched off, which leads to the differentiation of the remaining red cells. (**E**) In the absence of high proliferation, the cells rearrange to maximize the perimeter/area ratio. Characteristic penta- and hexagonal cell shapes emerge (cf. Supplementary file S6.4). Cells close to the boundary try to take a circular shape

where γ is a reactivity constant, a and b production constants and d the relative diffusion coefficient of ligand and receptor. We note that the equations correspond to the classical Schnakenberg-type Turing mechanism (Gierer and Meinhardt, 1972; Schnakenberg, 1979). It has previously been shown that such a receptor-ligand interaction can explain symmetry breaking in various morphogenetic systems (Badugu *et al.*, 2012; Cellière *et al.*, 2012; Menshykau and Iber, 2013; Menshykau *et al.*, 2012; Menshykau *et al.*, 2014; Tanaka and Iber, 2013).

Depending on the type of domain we observe different patterns. On a continuous domain, we obtain the well-known regular spot pattern (Fig. 6B). On an idealized static cellular domain an overall regular pattern with irregular internal structure (Fig. 6C) can be observed. Decreasing the simulation parameter γ, which inversely controls the distance between the spots, leads to even more unexpected patterns: for $\gamma = 100$, the local regularity is completely lost (Fig. 6D). Finally, on a dynamically growing cellular domain, where the local proliferation rate was set proportional to the R^2L signal, we obtain irregular patterns (Fig. 6E). For a lower value $\gamma = 100$, clusters of cells with

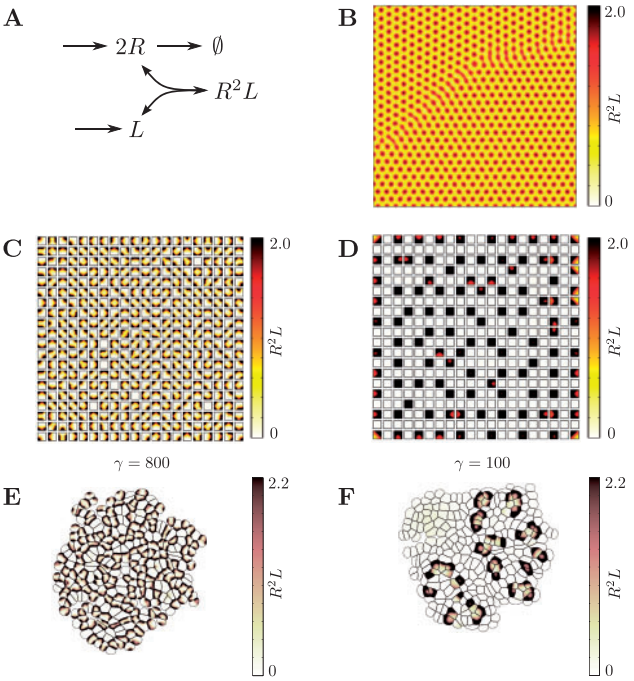

Fig. 6. Turing patterning on growing cellular domains. (**A**) Turing instability can be achieved by Schnakenberg-type reactions, involving a slowly diffusing compound R, here interpreted as a receptor, and a fast diffusing compound L, here interpreted as a freely diffusing ligand. One ligand molecule binds to two receptors, leading to the complex R^2L. The complex can be interpreted as a biological signal. (**B**) The model is solved on a continuous square lattice (using $d = 1$, $\gamma = 800$, $a = 0.1$, $b = 0.9$), resulting in the classical regular spot-pattern. The biological signal R^2L is shown. (**C**) The same system as in B is solved on an idealized static cellular domain, i.e. the diffusion of the receptor R is restricted to a cell. The emerging biological signal R^2L is now distributed irregularly. (**D**) The same system as in C, but with $\gamma = 100$, is solved on an idealized static cellular domain. Fewer cells show significant levels of signal R^2L and no regular pattern can be found (salt-and-pepper pattern). (**E**) The same system as in C is solved on a growing cellular domain. The proliferation rate of a cell is set proportional to its signal R^2L. The resulting pattern features regularity on a larger scale, but the local patterning significantly differs from the behaviour on continuous (**B**) and static cellular (**C**) domains. (**F**) The same system as in D is solved on growing cellular domain. The proliferation rate of a cell is set proportional to the local intensity of the signal R^2L. Clusters of active cells with high levels of R^2L emerge

high R^2L signalling levels emerge (Fig. 6F). In conclusion, even relatively simple signalling mechanisms can lead to significantly different results, depending on how the tissue is represented.

5 Discussion

We developed an extendible and open-source cell-based simulation environment, which is tailored to study morphogenetic problems. The novel framework permits the coupled simulation of a physically motivated viscoelastic cell model with regulatory signalling models. Processes such as viscous dissipation, elasticity, advection, diffusion, local reactions, local mass sources and sinks, cell division and cell differentiation are implemented. By applying our framework to Turing signalling systems, we show that the signalling systems may behave differently on dynamic tissues than on simple continuous tissue representations. We therefore advocate to test continuous morphogenetic signalling models on dynamically growing cellular domains.

The presented framework permits to study a variety of mechano-regulatory mechanisms. By making the cell division orientation dependent on signalling cues, the effect on the macroscopic tissue geometry may be studied. Cell migration can be modelled by introducing gradient-dependent forces on specific cell types. Cell sorting may be achieved by specifying multiple cell types with differential cell–cell junction strengths. The framework is specifically designed to study the mutual effects of signalling and biophysical cell properties.

The viscoelastic cell model represents cell shapes at high resolution and is thus, unlike the vertex model, not restricted to densely packed tissues. Furthermore, hydrodynamic interaction, membrane tension and hydrostatic pressure are integral components of the model. The fact that a velocity field is available on the entire domain is a critical advantage to account for advection of the signalling components, thus allowing for a spatial description of intracellular concentrations. The model is, however, not easily extendable to the third dimension. Because a meshing of the surface will be required, the algorithmic and computational complexity are expected to be significant and subject to future work. The presented framework is, however, ideal to study intrinsically 2D morphogenetic problems, such as apical surface dynamics of epithelia as studied previously also by Farhadifar *et al.* (2007) and Ishihara and Sugimura (2012) in 2D.

Funding

The authors acknowledge funding from the SNF Sinergia grant 'Developmental engineering of endochondral ossification from mesenchymal stem cells' and the SNF SystemsX RTD NeurostemX.

Conflict of Interest: none declared.

References

Badugu,A. *et al.* (2012) Digit patterning during limb development as a result of the BMP-receptor interaction. *Sci. Rep.*, **2**, 991.

Bellusci,S. *et al.* (1997) Involvement of sonic hedgehog (Shh) in mouse embryonic lung growth and morphogenesis. *Development*, **124**, 53–63.

Cellière,G. *et al.* (2012) Simulations demonstrate a simple network to be sufficient to control branch point selection, smooth muscle and vasculature formation during lung branching morphogenesis. *Biol. Open*, **1**, 775–788.

Chen,S. and Doolen,G.D. (1998) Lattice Boltzmann methods for fluid flows. *Annu. Rev. Fluid Mech.*, **30**, 329–364.

Chu,Y.-S. *et al.* (2005) Johnson-Kendall-Roberts theory applied to living cells. *Phys. Rev. Lett.*, **94**, 028102.

Drasdo,D. *et al.* (2007) On the role of physics in the growth and pattern formation of multi-cellular systems: what can we learn from individual-cell Based Models? *J. Stat. Phys.*, **128**, 287–345.

Farhadifar,R. *et al.* (2007) The influence of cell mechanics, cell-cell interactions, and proliferation on epithelial packing. *Curr. Biol.*, **17**, 2095–2104.

Feng,Z.-G. and Michaelides,E.E. (2004) The immersed boundary-lattice Boltzmann method for solving fluid particles interaction problems. *J. Comput. Phys.*, **195**, 602–628.

Forgacs,G. *et al.* (1998) Viscoelastic properties of living embryonic tissues: a quantitative study. *Biophys. J.*, **74**, 2227–2234.

Gierer,A. and Meinhardt,H. (1972) A theory of biological pattern formation. *Kybernetik*, **12**, 30–39.

Graner,F. and Glazier,J. (1992) Simulation of biological cell sorting using a two-dimensional extended Potts model. *Phys. Rev. Lett.*, **69**, 2013–2016.

Guo,Z. *et al.* (2002) A coupled lattice BGK model for the Boussinesq equations. *Int. J. Numer. Methods Fluids*, **39**, 325–342.

Hoehme,S. and Drasdo,D. (2010) A cell-based simulation software for multicellular systems. *Bioinformatics*, **26**, 2641–2642.

Ishihara,S. and Sugimura,K. (2012) Bayesian inference of force dynamics during morphogenesis. *J. Theor. Biol.*, **313**, 201–211.

Menshykau,D. and Iber,D. (2013) Kidney branching morphogenesis under the control of a ligand-receptor-based Turing mechanism. *Phys. Biol.*, **10**, 46003.

Menshykau,D. *et al.* (2012) Branch mode selection during early lung development. *PLoS Comput. Biol.*, **8**, e1002377.

Menshykau,D. *et al.* (2014) An interplay of geometry and signaling enables robust lung branching morphogenesis. *Development*, **141**, 4526–4536.

Merks,R.M.H. *et al.* (2011) VirtualLeaf: an open-source framework for cell-based modeling of plant tissue growth and development. *Plant Physiol.*, **155**, 656–666.

Minc,N. *et al.* (2011) Influence of cell geometry on division-plane positioning. *Cell*, **144**, 414–426.

Newman,T.J. (2005) Modeling Multicellular Systems Using Subcellular Elements. *Math. Biosci. Eng.*, **2**, 613–624.

Peskin,C.S. (2003) The immersed boundary method. *Acta Numerica,* **11**, 479–517.

Pitt-Francis,J. *et al.* (2009) Chaste: a test-driven approach to software development for biological modelling. *Comput. Phys. Commun.*, **180**, 2452–2471.

Ponce Dawson,S. *et al.* (1993) Lattice Boltzmann computations for reaction-diffusion equations. *J. Chem. Phys.*, **98**, 1514.

Rejniak,K.A. (2007) An immersed boundary framework for modelling the growth of individual cells: an application to the early tumour development. *J. Theor. Biol.*, **247**, 186–204.

Rejniak,K.A. *et al.* (2004) A computational model of the mechanics of growth of the villous trophoblast bilayer. *Bull. Math. Biol.*, **66**, 199–232.

Restrepo,S. *et al.* (2014) Coordination of patterning and growth by the morphogen DPP. *Curr. Biol.*, **24**, R245–R255.

Sandersius,S.A. *et al.* (2011a) A'chemotactic dipole' mechanism for large-scale vortex motion during primitive streak formation in the chick embryo. *Phys. Biol.*, **8**, 045008.

Sandersius,S.A. *et al.* (2011b) Emergent cell and tissue dynamics from subcellular modeling of active biomechanical processes. *Phys. Biol.*, **8**, 045007.

Schnakenberg,J. (1979) Simple chemical reaction systems with limit cycle behaviour. *J. Theor. Biol.*, **81**, 389–400.

Swat,M.H. *et al.* (2012) *Multi-Scale Modeling of Tissues Using CompuCell3D. Methods Cell Biol.*, 325–366.

Tanaka,S. and Iber,D. (2013) Inter-dependent tissue growth and Turing patterning in a model for long bone development. *Phys. Biol.*, **10**, 56009.

Turing,A.M. (1952) The chemical basis of morphogenesis. *Phil. Trans. R. Soc. B Biol. Sci.*, **237**, 37–72.

Zhang,J. *et al.* (2007). An immersed boundary lattice Boltzmann approach to simulate deformable liquid capsules and its application to microscopic blood flows. *Phys. Biol.*, **4**, 285–295.