Hindawi
Scientific Programming
Volume 2017, Article ID 8721612, 11 pages
https://doi.org/10.1155/2017/8721612

Research Article

Implementation of an Agent-Based Parallel Tissue Modelling Framework for the Intel MIC Architecture

Maciej Cytowski,[1] Zuzanna Szymańska,[1] Piotr Umiński,[2] Grzegorz Andrejczuk,[2] and Krzysztof Raszkowski[2]

[1]*ICM, University of Warsaw, Pawińskiego 5a, Warszawa, Poland*
[2]*Intel Technology Poland, Słowackiego 173, Gdańsk, Poland*

Correspondence should be addressed to Maciej Cytowski; m.cytowski@icm.edu.pl

Received 17 October 2016; Revised 23 December 2016; Accepted 1 February 2017; Published 23 February 2017

Academic Editor: Raphaël Couturier

Timothy is a novel large scale modelling framework that allows simulating of biological processes involving different cellular colonies growing and interacting with variable environment. *Timothy* was designed for execution on massively parallel High Performance Computing (HPC) systems. The high parallel scalability of the implementation allows for simulations of up to 10^9 individual cells (i.e., simulations at tissue spatial scales of up to 1 cm^3 in size). With the recent advancements of the *Timothy* model, it has become critical to ensure appropriate performance level on emerging HPC architectures. For instance, the introduction of blood vessels supplying nutrients to the tissue is a very important step towards realistic simulations of complex biological processes, but it greatly increased the computational complexity of the model. In this paper, we describe the process of modernization of the application in order to achieve high computational performance on HPC hybrid systems based on modern Intel® MIC architecture. Experimental results on the Intel Xeon Phi™ coprocessor x100 and the Intel Xeon Phi processor x200 are presented.

1. Introduction

Mathematical modelling in biology aims at the representation of biological processes using a variety of mathematical and computational techniques. Addressing the most ambitious challenges of modern biology alongside new experimental studies very often requires analytical description and efficient "in silico" simulations. Nowadays, the use of mathematical language is widely recognized as a complementary tool for deepening the understanding of complex biological systems. The approach of mathematical modelling takes on a special value when computer simulations and predictions are compared with experimental results and should be seen as an iterative process, where experiments and computer simulations refine and improve one another by providing feedback. Apart from the cost aspect, mathematical modelling may involve specific aspects and phenomena which cannot be included in the experimental laboratory conditions. As a result, new hypotheses on biological mechanisms may come out. Thus, mathematical modelling in biology has a real chance of making a great impact on the development of biology, mathematics, and computational sciences.

The utility of mathematical models in biology is often limited by very high computational complexity. A typical example, where limitations associated with computational complexity play an important role, is simulations of development of cellular colonies of different kinds. Such phenomena are usually modelled with the use of the so-called individual-based models that are sometimes also referred to as agent-based models. In this approach, individual cells are explicitly modelled as single entities or agents and their properties are determined and can evolve according to a set of biophysical rules. The computational complexity of such models increases rapidly with the level of biological detail and amount of data included in the model. Just how important such models have become is indicated by the spectrum of their applications; that is, these models are widely used in a broad class of problems of immense significance which include, among others, issues such as the growth of bacterial colonies, solid tumors, embryogenesis, and wound healing.

There are several agent-based modelling tools available in the field of mathematical biology. Most of these tools have their own specific features related to modelling approach, usability, implementation, or performance. From the modelling perspective, there are worth mentioning tools that implement different variants of the Cellular Potts Model, *CompuCell3d* [1], *Morpheus* [2], and *Simmune* [3], off-lattice center-based models, *CellSys* [4], *PhysiCell* [5], *Biocellion* [6], and *Timothy*, or more universal tools which incorporate different models within a code library like *Chaste* [7]. Several tools provide improved user interface based on scripting language *(CompuCell3d)*, domain specific language *(Morpheus),* or advanced graphical user interface *(Simmune*, *Morpheus)*. An interesting usage model was introduced in *Chaste*, which is a library consisting of implementation of different modelling approaches. Users can define their own models by choosing and combining modules available in *Chaste*. In terms of performance, several tools provide single node parallelization based on OpenMP (*Morpheus*, *CellSys*, *PhysiCell*, and *CompuCell3d*) which enables computations to scale up to few millions cells. To our knowledge, *Timothy* was the first tool to introduce parallelization between nodes, which enabled large scale simulations on the order of several billions of cells. However, recently, another tool called *Biocellion* was made available. Both tools enable simulations to be carried out with the use of large core counts on a broad class of supercomputers.

Timothy is an open-source software distributed under the GNU GPL license and freely available at http://timothy.icm.edu.pl. With recent advancements in HPC and the development of new algorithms, the *Timothy* framework enables simulation of cellular colonies dynamics consisting of as much as 10^9 individual cells. In many eukaryotic systems, this corresponds to the size of the simulated structures to be as big as $1\,cm^3$ or larger. At this size, cellular colonies are palpable and the model has the potential to be of prognostic value to clinicians interested in a range of pathological situations. The mathematical model represented by the *Timothy* framework was already described in [8–10]. In Section 2, we recall only the most important characteristics of the model.

Timothy was already used to simulate various biological processes and it is still subject to continuous improvements. In order to demonstrate the computational complexity of simulations with very large number of cells, we present some of the details of one of the extreme scale tumor growth simulations that we executed on the IBM Power 775 system. The simulation consisted of a healthy 3D tissue composed of $2.5 * 10^8$ cells and a single tumor cell placed in the middle. As simulation proceeded, the tumor cells spread in the tissue. The cross section of one of the intermediate steps of the simulation with approximately 75,000 tumor cells is shown in Figure 3. A single time step of the simulation was computed in approximately 100 seconds with the use of 128 cores of the IBM Power 775 system. We recommend that a single time step should correspond to at most 3600 seconds of a real biological process development. This means that a simulation illustrating the three-month development of the geometric structure of the tumor would take 60 days of computing on the selected partition of the IBM Power 775 system.

Recently, the *Timothy* model has been extended to include blood vessels supplying nutrients to the tissue. Incorporating new features and functionalities into the model has enormous potential for future development and can take the multiscale mathematical modelling of cellular systems to a completely new level. However, in order to enable such development, appropriate steps need to be taken to ensure good performance on the emerging HPC architectures. One of the most important processor architectures nowadays is the Intel Many Integrated Core architecture (the Intel MIC architecture) introduced by Intel in 2010 with the prototype product codenamed Knights Ferry. The Intel MIC architecture together with recent Intel products is described in Section 3. The usefulness of this architecture for increasing the efficiency of computing applications has been already shown in number of publications [11, 12].

This paper describes the process of modernization of the *Timothy* computational model for the hybrid systems based on modern Intel Xeon Phi product family devices, which include Intel Xeon Phi coprocessors x100 (code name: Knights Corner (KNC)) and Intel Xeon Phi processors x200 (code name: Knights Landing (KNL)). In Section 4, we describe all necessary transition steps which include profiling, choosing an appropriate execution model, and reimplementation of *Timothy.* In Section 5, we present performance results and comparisons.

2. Mathematical Model Description

Cells are modelled as spheres that reside in three-dimensional space. The position of each cell is given by Cartesian coordinates of the sphere's center and, most importantly, is not limited to predefined lattice nodes (off-lattice model). Cells can interact with each other if located close enough. Each cell's neighborhood is defined by a cutoff distance.

Thanks to an individual-based modelling approach, the *Timothy* framework allows for introduction of intracellular processes in each individual cell [13]. One of the processes which is being modelled at this scale is the cell cycle, an ordered series of reactions leading to duplications of cell's biological contents and finally to the division into two new daughter cells (see Figure 1).

The interactions between two cells are described and computed by a modified Hertz model which includes attractive interactions related to adhesive forces and repulsive interactions associated with limited compressibility of cells [14]. For each cell, we consider such interactions with all cells in its neighborhood. Besides cells in the neighborhood, the external factor influencing the dynamics of a cell is its environment. Some cells can even migrate towards the areas where they have better living conditions. This phenomena, i.e., directed movement of cells along the concentration gradient of chemoattractant, is called chemotaxis. In other words, the environment may determine the direction of movement of the cell. The model takes into account this phenomenon including the chemotactic term in the equation describing the forces acting on a cell (see Figure 2). To sum

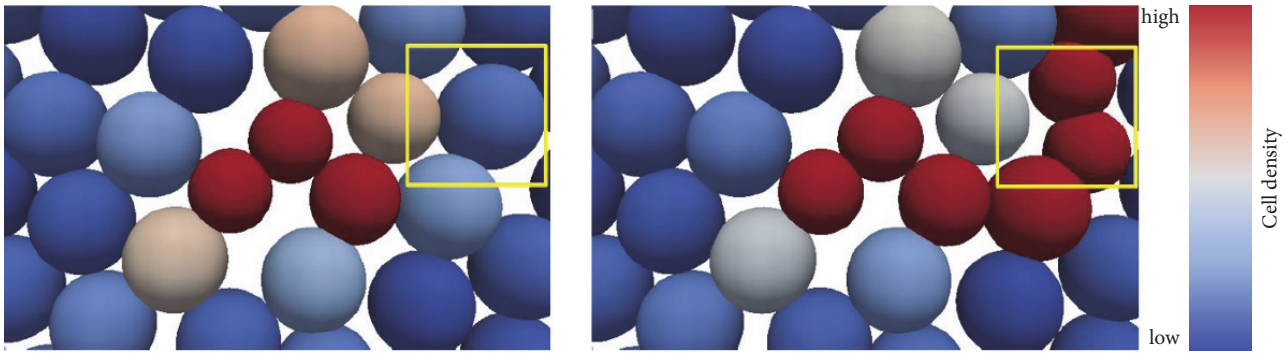

Figure 1: Mitosis of a given cell (indicated by a yellow frame). Colors indicate neighborhood-based cell density.

Figure 2: Schematic representation of forces influencing the cell movement.

up, we assume that the movement of a cell is influenced by the strengths of adhesion, the strengths of repulsion, and the phenomenon of chemotaxis.

On the other hand, the environment in which a cell lives also affects its internal dynamics, particularly its metabolism. A living cell constantly samples its environment checking if the conditions are appropriate and the level of nutrients is sufficient. This ability is particularly important when a cell has to decide on a possible mitosis. If the living conditions are sufficiently good, the cell progresses through the cell cycle and ultimately divides into two daughter cells. If environment conditions become harsh, it might not be able to pass through the cell cycle and becomes quiescent. A further deterioration of the environment can lead to cell death. The general equation describing the dynamics of global fields reads as follows:

$$\frac{\partial}{\partial t} Q(x,t) = D_Q \nabla^2 Q(x,t) - G(x,t) H(x,t), \quad (1)$$

where $Q(x,t)$ denotes the substance concentration, D_Q denotes the diffusion constant, and function $H(x,t)$ denotes the substance source whereas $G(x,t)$ denotes the substance uptake. The equation is equipped with Dirichlet boundary conditions. Both functions $H(x,t)$ and $G(x,t)$ are calculated at each node of the discretization grid (i.e., their values dependent on the concentration of cells and other factors such as concentration of blood vessels in surrounding tissue). We assume that discretization grid size used for solving substance equations is always larger than the cell diameter. In each time step, the concentration of cells in each grid cell is computed. Based on those values, the uptake function $G(x,t)$ is computed in each grid node. Then, the concentration of a given substance is interpolated to each biological cell with the cloud-in-cell method (CIC). Additionally, in cases such as simulating the dynamics of tissue with blood vessel, the source function $H(x,t)$ is also computed based on vessels concentration computed in each grid node. This way, the coupling between the discrete cellular and continuous environment descriptions is defined in our model. It should be strongly emphasized that we are dealing with the so-called hybrid approach, where the environment, which may be nutrients, metabolic wastes, temperature, and so on, is modelled with the use of continuous mathematical equations (partial differential equations), while cells in cellular colonies are modelled as discrete entities. Such hybrid approach has a few very important consequences on the implementation which will be mentioned in the next section.

The *Timothy* framework was developed to simulate multiple biological scenarios. For this purpose, it was designed so that adding new functionalities is straightforward in order to make the applicability of the model even greater. An example of such an extension is taking into account the blood vessels that is necessary in many biological and medical applications. In the model, blood vessels are introduced by specifying spheres whose collection forms the desired vessels shapes. These spheres represent cells that build an outer layer of blood vessels. In case of capillaries, they simply correspond to endothelial cells building up the vessel. Depending on the simulated scenario, each of these spheres can constitute a source of oxygen or other external nutrients. Suitable shapes of the blood vessels can be introduced into the model in two ways. One is to use an additional *Timothy* module which produces the collection of spheres for vessels defined by Bezier curves. This method is appropriate when the goal is to describe the capillaries. An alternative option is to import the actual data derived from medical imaging. Such data has to be properly prepared to run the simulation. For this purpose, one can use the *VisNow* package, which is an open access software developed at the Interdisciplinary Centre for Mathematical and Computational Modelling (ICM, University of Warsaw) and allows complex visual analysis and segmentation of the geometry to be studied. On the basis of the chosen geometry, one can provide digitized input data for the model. Figure 4 presents an example of a blood vessel geometry imported from microtomography together with a visualization of the concentration gradient of oxygen released by the vessel to the surrounding tissue.

Moreover, the *Timothy* framework allows for the description of three important biological scales of interest: the intracellular, the cellular, and the tissue scales. The multiscale

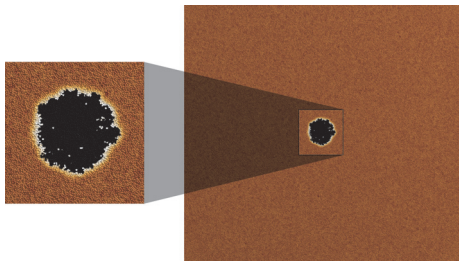

Figure 3: A tissue cross section consisting of approximately $2.5 \cdot 10^8$ healthy cells and around 75,000 tumor cells. The tumor cells are shown in black. On the left-hand side, the structure of the tumor is shown in a close-up view.

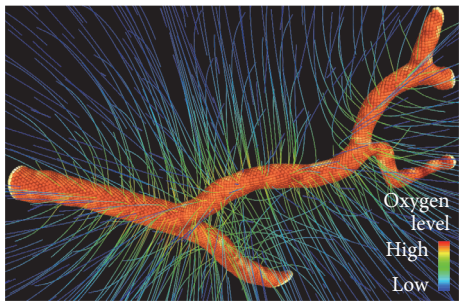

Figure 4: The *Timothy* framework has been recently extended to include blood vessels that supply nutrients to the tissue. This image originates from one of the first simulations of such type. It shows a vessel with streamlines of the concentration gradient of oxygen in the surrounding tissue. Realistic vessel shapes were obtained through processing of medical imaging data with the VisNow package [15].

approach makes it possible to better understand the links between processes occurring inside the cell and phenomena observed at the macroscopic level. This concerns mainly the so-called signaling and metabolic pathways which are the cascades of biochemical reactions responsible for (i) maintenance of the intracellular homeostasis, (ii) targeted response of cells to external stimuli, and (iii) proper intracellular metabolism. Abnormal changes in these pathways are often the origin of serious diseases such as inborn and acquired errors of metabolism or cancer. The latter applies particularly to carcinogenic mutations in genes regulating cell division and death. To link the aforementioned intracellular processes with the clinically observed phenotype is the essential difficulty related to understanding biological processes. In the future, computational frameworks like *Timothy* may have a genuine impact on our understanding of the courses of many diseases and in optimizing and adjusting treatments to meet the needs of each patient.

To give the reader a clear understanding of what types of simulations can be performed by *Timothy*, we present the visualizations of sample results in Figures 3 and 4.

3. Computing Architecture Overview

In our experiments, we used two generations of Intel Xeon Phi processors/coprocessors. The Intel Many Integrated Core architecture (Intel MIC architecture), used in the Intel Xeon Phi product family, is complementary to the traditional many-core architecture represented by multiple generations of Intel Xeon® processors. It targets traditional HPC workloads as well as other highly parallel, computation intensive tasks such as machine learning. We will describe briefly details of this architecture to help aid the reader in understanding the obtained performance results.

The first product from the Intel Xeon Phi family, the Intel Xeon Phi coprocessor x100 (code name: Knights Corner (KNC)), is available on a separate extension board [16]. It contains up to 61 cores, each of them capable of running four hardware threads, each core has its own L2 cache and a high bandwidth bidirectional ring connecting them together. All L2 caches are fully coherent across the entire processor. Memory controllers are connected to the same ring and provide interfaces to the on-board DDR memory. The cores are able to execute a proprietary 512-bit vector instruction set. A standard PCIe interface is used to communicate with the host processor. The Intel Xeon Phi coprocessor x100 runs a customized distribution of Linux OS and can be used in hosts running either Linux or Microsoft Windows OS. The coprocessor is not binary compatible with other Intel processors, so code recompilation is required.

The second-generation Intel Xeon Phi Product Family x200 (code name: Knights Landing (KNL)) is available as a stand-alone processor as well as a coprocessor [17]. Intel Xeon Phi x200 devices have up to 72 cores, each core running four hardware threads. Internally, the processor contains up to 36 tiles, each of which contains two improved Silvermont cores. Each core is equipped with L1 data and instruction caches and two additional vector processing units (VPU) that execute standard AVX-512 vector instructions. Each VPU is capable of executing floating point operations on vectors of 16 double-precision or 32 single-precision numbers. Two cores that reside on the same tile share 1 MB of L2 cache. All caches are fully coherent. Tiles are connected with a high-throughput mesh interconnection.

The Intel Xeon Phi processor x200 is binary compatible with other Intel Xeon processors. AVX-512 vector extension instructions are part of the standard Intel architecture instruction set and will be supported by other Intel processors.

The Intel Xeon Phi processor x200 can address up to 384 GB of standard DDR memory. In addition to that, it has 16 GB of high bandwidth, on-package MCDRAM memory. The MCDRAM memory offers much higher bandwidth than the traditional DDR memory (up to 480 GB/s on STREAMS Triad benchmark, compared to ~90 GB/s on DDR for the same benchmark), with only a slightly higher latency. Therefore, it offers a significant performance boost for parallel workloads. The MCDRAM memory can be used in two modes: cache and flat. In the flat mode, the MCDRAM memory can be addressed directly by cores. To distinguish between DDR and MCDRAM, the two kinds of memory are exposed by the OS as separate NUMA nodes. Software decides whether a given variable is in MCDRAM or DDR. In the cache mode, the MCDRAM memory is used as a large L3 cache managed entirely by hardware.

4. Implementation Details

The main emphasis in the design and development of *Timothy* was put on efficient parallelization and achieving high computational performance. The framework is currently composed of two modules: the cellular dynamics module and the nutrients environment module. Both modules are of a different computational nature, with the former being implementation of a specific type of discrete agent-based model while the latter being responsible for discretizing and solving a continuous mathematical system of PDEs. Moreover, parallelization of both modules is based on domain decomposition methods of different kind: the cellular dynamics module uses a geometrical Peano-Hilbert decomposition method while the nutrients environment module uses block decomposition. This makes parallel data sharing scheme a bit more complicated and harder to implement but by using the asynchronous communication mechanism we are able to maintain very good scalability of both modules. Both modules are coupled together; that is, discrete objects can consume and/or produce specific nutrients which must be reflected in variability of the cellular environment.

Figure 5 presents computational scheme of simulations in *Timothy*. The biological cellular systems are simulated by computing their state at successive time steps over a specified time span. In each iteration, every process performs computations of both modules.

Computations of the cellular dynamics module consist of the following steps:

(i) Domain decomposition: An algorithm based on Peano-Hilbert space filling curves finds the optimal distribution of ownership of cells between available processes. For each cell based on cell's location, the algorithm first finds its corresponding values in the [0, 1] interval by using an inverse mapping associated with the curve. The [0, 1] interval is then cut into fragments containing equal numbers of cells. Those fragments (subsets containing cells) are finally mapped to corresponding parallel processes. This process is schematically presented for a 2-dimensional case in Figure 6. At the end of this step domain decomposition is executed (i.e., cells are migrated between processes according to mappings).

(ii) Tree build: Cells assigned to each process are organized in an octree structure based on their geometrical localization. The procedure starts with a root node containing all local cells. The root node is repeatedly subdivided into eight octants. The procedure stops when each leaf node of the tree contains at most a single cell.

(iii) Cells exchange init: The exchange areas (boundaries) are found for each pair of processes. Data exchange is initiated with an asynchronous MPI call.

(iv) Computing potential (local data): Potential component is computed by octree traversal for each cell based on locally available data.

(v) Cells exchange wait: Data exchange is finished.

(vi) Computing potential (received data): Potential component is computed by octree traversal for each cell based on data exchanged with neighboring processes.

(vii) Potential exchange init: Potential values exchange is initiated with an asynchronous MPI call.

(viii) Computing gradient of the potential (local data): Gradient of the potential component is computed by octree traversal for each cell based on locally available potential values.

(ix) Potential exchange wait: Potential values exchange is finished.

(x) Computing gradient of the potential (received data): Gradient of the potential component is computed by octree traversal for each cell based on potential values exchanged with neighboring processes.

Computations of the nutrients environment module consist of the following steps:

(i) Solving nutrients environment PDEs: The *Hypre* library [18] is used for solving nutrients environment PDEs. The discretization of PDEs is achieved with the use of an implicit in time finite difference scheme. The resulting system is then defined in the ParCSR parallel format. The data decomposition is achieved by assigning computational grid blocks to different processes. Conjugate gradient solver preconditioned by the BoomerAMG algebraic multigrid method is used to solve the system.

(ii) Computing nutrients' gradients: Gradients of given nutrients are computed. Resulting vector field can be used to simulate the chemotactic movement of cells.

At the end of each iteration of the main simulation loop, the cells' states are being updated. Cells can grow, change their current cell cycle phase, or move to a different location in a 3D space.

From the technical point of view, *Timothy* is implemented in the C programming language and is parallelized with a hybrid model which combines MPI and OpenMP. It uses MPI to communicate among nodes and uses OpenMP-based shared memory programming in each node. Shared memory parallelization is used in the most compute intensive parts of each MPI process (i.e., in computing the potential and the gradient of the potential steps as well as in solving nutrients environment PDEs step, where we make use of MPI/OpenMP parallelization available in the *Hypre* library). *Timothy* uses the MPI parallel I/O scheme to handle program output and the checkpoint/restart mechanism. It has several dependencies: the *Zoltan* library [19] used for domain decomposition, the *Hypre* library [18] used for solving partial differential equations describing cellular environment, and the *SPRNG* 2.0 library [20] for random number generation.

Timothy was tested on various massively parallel platforms including IBM Blue Gene/Q, IBM Power 775, and Cray XT40. The scalability tests executed on those platforms confirm that *Timothy* is capable of performing extreme scale simulations of individual-based models of cellular biosystems.

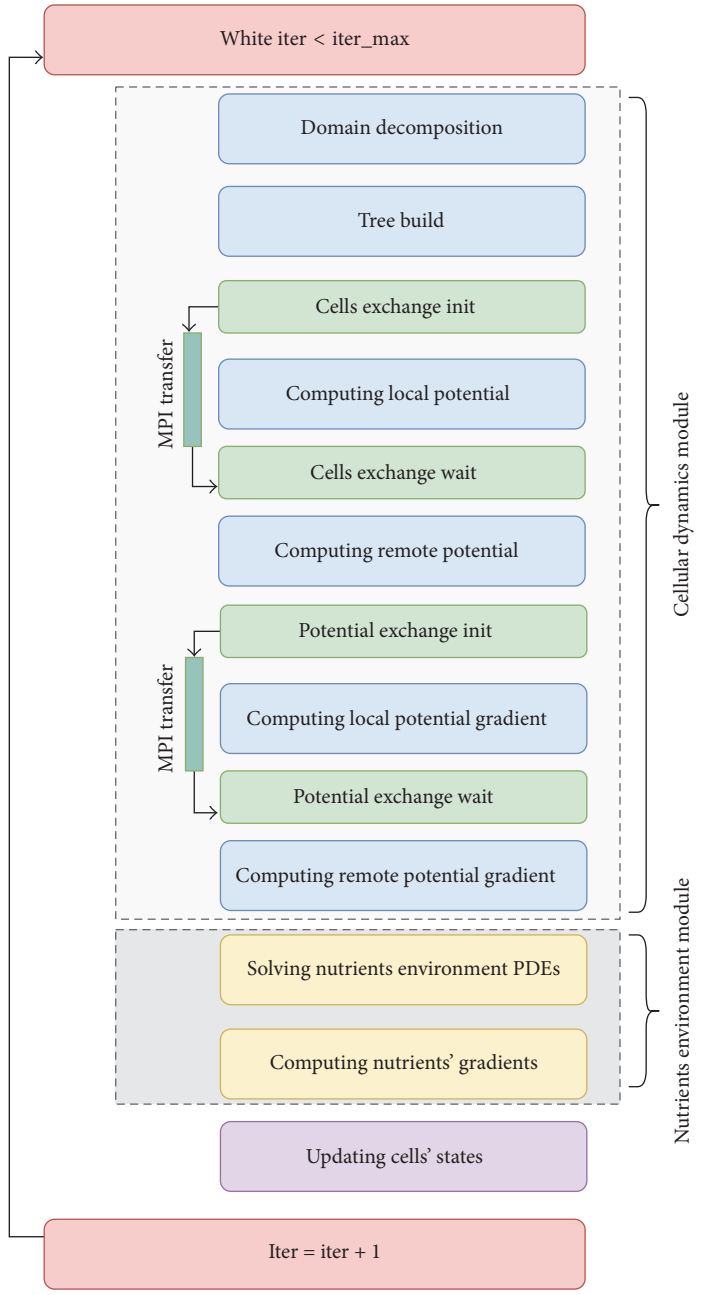

Figure 5: Scheme of simulation stages in *Timothy*.

5. Results and Discussion

The profiling of the application indicates that there are two main computational kernels, one corresponding to the cellular dynamics module and the other one corresponding to the nutrients environment module.

The cellular dynamics module presents very good OpenMP scalability, as can be seen in Table 1 (timings for potential and potential gradient computations are presented). This part of the application was our candidate for achieving good performance on the Intel MIC architecture.

5.1. Timothy Modernization for the Offloading Model. *Timothy* modernization for the Intel MIC architecture was performed using offloading model. Under this approach, the highly parallel part of the code is executed on the Intel Xeon Phi coprocessor/processor, while less-parallel part is executed on the traditional host processor. The original source code is decorated by the pragma-based compiler directives and the compiler with the corresponding run-time libraries arranges the code execution and data transfer between the host processor and the offloading target processors/coprocessors.

Table 1: OpenMP scalability of two stages of simulations in *Timothy*.

Threads per node	Computing potential seconds (speedup)	Computing gradient seconds (speedup)
1	57.75	64.02
2	29.91 (1.93x)	37.73 (1.70x)
4	16.27 (3.55x)	22.55 (2.84x)
8	8.64 (6.68x)	12.11 (5.29x)
16	4.85 (11.91x)	6.59 (9.71x)
32 (SMT2)	2.90 (19.91x)	3.88 (16.5x)
64 (SMT4)	1.99 (29.02x)	2.78 (23.03x)

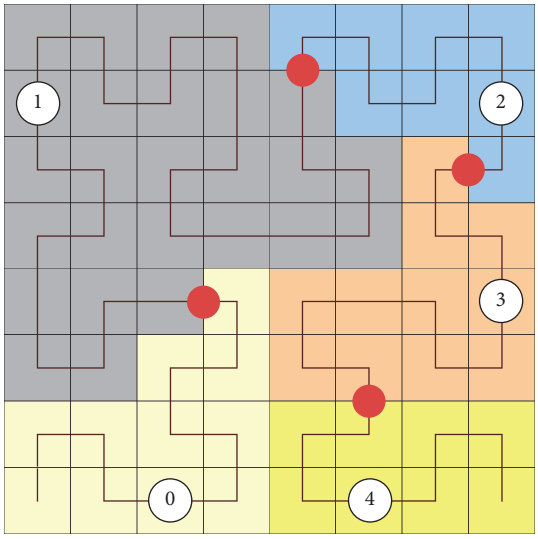

Figure 6: Example mapping for a 2-dimensional case and 5 parallel processes achieved with the use of Peano-Hilbert algorithm.

Modernization of *Timothy* was based on an important observation that the synchronization (exchange of information) between two modules can be performed at the end of each iteration. Therefore, it is possible to execute both modules simultaneously.

As it is shown in Figure 7, we decided to redesign the application in such a way that the nutrients environment module is computed on the host, while the cellular dynamics module is executed on the coprocessor.

We performed the following porting steps:

(1) New version of tree build and octree traversal algorithms: The previous implementation of tree build and octree traversal algorithms was based on C pointers which turned out to be very hard to implement with the offload mechanism. A new version was created, which operates on tables and indexes. Second important modification was related to the implementation of a nonrecursive (iterative) octree traversal algorithm, which proved to be much more efficient with higher numbers of threads.

(2) Offload scheme implementation: An offload and data transfer scheme was created and implemented with the use of the pragma-based offload mechanism available in the Intel compilers suite.

(3) Scalability test of the computational kernel on the offloading target: Various available work distribution types and thread affinity models were tested on the Intel MIC architecture. The most efficient setup was achieved with the use of *schedule(dynamic, 64)* and *granularity=thread,compact* settings. We achieved more than 157x speedup with the use of all 244 threads of the Intel Xeon Phi coprocessor x100.

(4) Implementation of an efficient overlapping scheme: We used the *signal-wait* mechanism to synchronize work between the host and the offloading target.

(5) Multiple MPI processes per node: The application was prepared to be executed with multiple MPI processes on the host. Each MPI process can offload work to the offloading target.

It should be emphasized that, in the resulting application, tree building is executed on the host side and only the resulting table representing the tree structure is being transferred to the offloading target. Therefore, only the octree traversal type algorithms are executed on the coprocessor.

The performance benchmark of the resulting hybrid application was composed of more than 5 million cells and the environment PDEs discretization grid size of $220 \times 220 \times 220$.

5.2. Experimental Results on Knights Corner. First experiments were performed using Intel Xeon Phi coprocessors x100 (Knights Corner). The computing system used in development and testing phase was composed of the single computational node with two Intel Xeon processors E5-2699 v3 @2.30 GHz (host) and two Intel Xeon Phi coprocessors 7120P (offloading targets). It ran Red Hat Enterprise Linux* version 7.2 and the Intel Many Core Software Stack (Intel MPSS) version 3.7.2.

The best results for the accelerated version of the code were achieved on the host with the use of 8 MPI processes, each with 9 OpenMP threads (host consists of $2 * 18 = 36$ physical cores, i.e., 72 hyperthreads). We executed the new accelerated version of *Timothy* on one and two coprocessors. The host/target communication was performed over PCIe

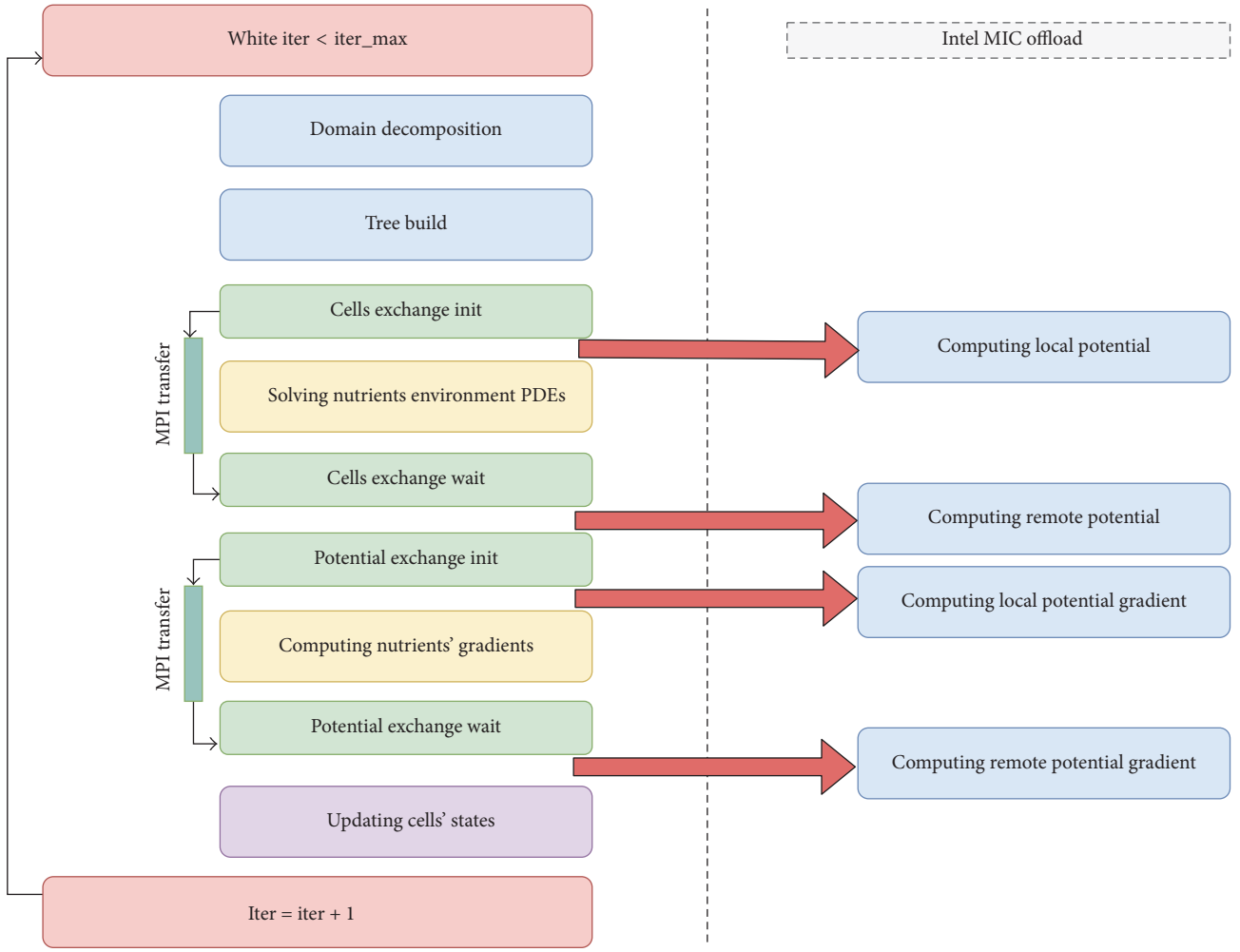

Figure 7: Scheme of simulation stages in *Timothy*. Stages executed on coprocessors were indicated on the right-hand side.

Table 2: *Timothy*: execution on the Intel Xeon Phi x100 coprocessor.

Configuration	Time [s]
Host + 1 KNC coprocessor	180
Host + 2 KNC coprocessors	135

(Gen2) interface. Results are listed in Table 2. The performance on a single host + 2 coprocessors is 25% better than on a single host and a single coprocessor.

5.3. Octree Benchmark Results on Knights Landing. In order to evaluate the expected performance results of the Intel Xeon Phi processor x200, a simple application benchmark called *Octree* was created. The benchmark was made available on the *Timothy* web page. The benchmark program has a single argument which is the number of discrete elements (agents). In the first step, the memory is allocated and positions of discrete elements are randomly chosen with the use of the Box-Muller transform (normal distribution). The benchmark runs on a single node and consists of two phases: tree generation for a given set of discrete elements and tree walk. Tree generation is sequential while tree walking is a highly parallel task. OpenMP is used to spread the workload into multiple threads. The main purpose of running the *Octree* benchmark was to evaluate how *Timothy* and other agent-based modelling codes could benefit from using the new architecture.

Octree does not use any architecture-specific enhancements; in particular, it was not manually tuned for the Intel Xeon Phi processor x200. The Intel C++ Compiler was used to compile *Octree* for the Intel Xeon Phi and for Intel Core™ processors.

We executed Octree on Intel Xeon Phi processor 7250, which runs at 1.40 GHz and has 68 cores and 16 GB MCDRAM. The benchmark was executed for 10 million discrete elements. The MCDRAM memory was configured in the flat mode. The benchmark was executed twice: using the DDR memory and using the MCDRAM mode. In the latter case, entire data structures fit into MCDRAM. Selection of the memory was done at run time using standard *numactl* command to select NUMA node(s) for the memory allocation, so

Table 3: *Octree* execution: tree generation and tree walk time [s].

	Tree generation			Tree walk		
	KNL + DDR	KNL + MCDRAM	Intel Core	KNL + DDR	KNL + MCDRAM	Intel Core
Run 1	10.4275	12.0853	3.2020	3.7581	2.0220	11.0934
Run 2	10.7856	12.3703	3.3896	3.6044	1.8730	11.0872
Run 3	10.8040	12.4549	3.3644	3.6214	1.8719	11.0802
Run 4	10.7859	12.3930	3.3833	3.6012	1.8704	11.0865
Average	*10.79183*	*12.32588*	*3.33483*	*3.64628*	*1.90933*	*11.08683*

the benchmark code remained unchanged. Each benchmark execution performed the building of a tree form, repeated independently 4 times. Table 3 shows the tree generation and tree walk times on the Intel Xeon Phi processor x200 (KNL) and corresponding results on the Intel Core processor (Intel Core i7-5960X processor @ 3.00 GHz).

As expected, the execution of sequential tree generation was almost 3 times slower on KNL than on the reference Intel Core processor. Note also that execution of this phase using the MCDRAM memory was about 14% slower than on DDR due to its slightly higher latency. Also as expected, a vast speedup was obtained on tree walk. By using MCDRAM and multiple threads, we obtained almost 6 times better results than on the reference Intel Core (3 times better by using multiple threads and DDR only).

5.4. Timothy on Knights Landing. Results presented in the previous section showed that using next generation of the Intel Xeon Phi processor we may get additional performance boost. Therefore, the entire *Timothy* application was run on KNL. For this experiment, we used the same computation node as in Section 5.2 (2x Intel Xeon processors E5-2699 v3 @2.30 GHz with 18 cores each). We also used two stand-alone KNL computation nodes, each with Intel Xeon Phi processors 7210 (64 cores). 16 GB of the MCDRAM memory on each KNL node was configured in flat mode.

For offloading from the host to the KNL offloading targets, we used the offloading over fabric functionality, available as a part of the Intel Xeon Phi Processor Software package [21]. It utilizes fabric interface (like Intel Omni-Path Architecture™ or InfiniBand) to offer the same offloading functionality available for the Intel Xeon Phi coprocessors connected over PCIe. In our experiments, we used Intel Omni-Path Architecture interfaces and the Intel OPA 10.2.0.0.158 software.

The same parameters of the simulation model as in Section 5.2 were used during execution on KNL nodes. Also like in experiments described in Section 5.2, the best results were obtained when 8 MPI ranks were executed on the offloading host. The MCDRAM memory was used on the offloading targets. *Timothy* source code modernized for KNC was not modified for KNL. The offloading code was recompiled using the Intel icc compiler 17.0.0 that supports AVX-512 instructions. Offloading over fabric run time was responsible for handling architectural differences, like MCDRAM access or communication over fabric interface. Table 4 shows the results of *Timothy* execution on one host and one and two KNL offloading target nodes. The performance on a single host + 2 KNL nodes was 22% better than on a single host and a single KNL node.

Table 4: *Timothy*: execution on the Intel Xeon Phi x200 processor.

Configuration	Time [s]
Host + 1 KNL processor	135
Host + 2 KNL processors	105

Finally, we ran complete *Timothy* on a single KNL node. The source code was recompiled with offloading functions disabled. Multiple configuration of the computation node and different number of MPI ranks were tried. The best performance was achieved with 8 MPI ranks, each running 34 OpenMP threads. MCDRAM was used in flat mode. On such configuration, the model used in previous experiments was calculated in 208 seconds (54% worse compared to a single host + 1 KNL node).

6. Conclusions

Mathematical modelling has a tremendous impact on our understanding of complex biological systems and is very often used as a research tool complementary to the traditional, heuristic experimental approach. We believe that the *Timothy* model might have a unique benefit for the scientific community, since it is the first and, to our knowledge, the only available individual-based cellular biosystem modelling tool capable of simulating processes at the tissue scale.

In this paper, we described the process of porting *Timothy* to the Intel MIC architecture. In case of the Intel Xeon Phi coprocessor x100 (KNC), we showed how the application was redesigned for execution on a hybrid system consisting of both a host processor and a coprocessor and presented the resulting performance gain. Secondly, we created a benchmark program to test the performance of accelerated parts of *Timothy* on the new Intel Xeon Phi processor x200. One of the main issues we looked at was the performance gain from using the MCDRAM memory for computations. Finally, we ran *Timothy* on the new Intel Xeon Phi processor x200 (KNL).

The results of experiments show that a system consisting of a single Intel Xeon processor-based host and a single KNL offloading target connected over fast fabric interface provides the same results as a system with an identical Intel Xeon processor and two KNC coprocessors connected over PCIe.

Despite the mixed nature of the workload (highly parallel code mixed with more sequential code), performance on a single KNL node (without host) was also very high.

The obtained results show the efficient migration paths for applications already ported to the older Intel Xeon Phi coprocessor x100 towards the new Intel Xeon Phi processors x200.

The tree walk algorithm which is part of *Timothy* and was implemented in the *Octree* benchmark is widely used in many different agent-based modelling tools (e.g., in computational cosmology, social sciences, or epidemiology). Many interesting usage examples are listed in the review publications like [22] and books [23]. Therefore, we believe that results presented in this paper are very promising and will have much broader impact on computational sciences.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Maciej Cytowski and Zuzanna Szymańska were partially supported by The National Centre for Research and Development Grant STRATEGMED1/233224/10/NCBR/2014. Zuzanna Szymańska acknowledges the National Science Centre Poland Grant 2014/15/B/ST6/05082. Maciej Cytowski was partially supported and acknowledges the MICLAB Project no. POIG.02.03.00.24-093/13 cofinanced by the European Regional Development Fund under the Innovative Economy Operational Programme. The authors thank Mateusz Mowka for language review of the manuscript.

References

[1] M. H. Swat, G. L. Thomas, J. M. Belmonte, A. Shirinifard, D. Hmeljak, and J. A. Glazier, "Multi-scale modeling of tissues using CompuCell3D," *Methods in Cell Biology*, vol. 110, pp. 325–366, 2012.

[2] J. Starruß, W. de Back, L. Brusch, and A. Deutsch, "Morpheus: a user-friendly modeling environment for multiscale and multicellular systems biology," *Bioinformatics*, vol. 30, no. 9, pp. 1331–1332, 2014.

[3] B. R. Angermann, F. Klauschen, A. D. Garcia et al., "Computational modeling of cellular signaling processes embedded into dynamic spatial contexts," *Nature Methods*, vol. 9, pp. 283–289, 2012.

[4] S. Hoehme and D. Drasdo, "A cell-based simulation software for multi-cellular systems," *Bioinformatics*, vol. 26, no. 20, pp. 2641–2642, 2010.

[5] P. Macklin, M. E. Edgerton, A. M. Thompson, and V. Cristini, "Patient-calibrated agent-based modelling of ductal carcinoma in situ (DCIS): from microscopic measurements to macroscopic predictions of clinical progression," *Journal of Theoretical Biology*, vol. 301, pp. 122–140, 2012.

[6] S. Kang, S. Kahan, J. McDermott, N. Flann, and I. Shmulevich, "Biocellion: accelerating computer simulation of multicellular biological system models," *Bioinformatics*, vol. 30, no. 21, pp. 3101–3108, 2014.

[7] G. R. Mirams, C. J. Arthurs, M. O. Bernabeu et al., "Chaste: an open source C++ library for computational physiology and biology," *PLoS Computational Biology*, vol. 9, no. 3, Article ID e1002970, 2013.

[8] M. Cytowski and Z. Szymanska, "Large scale parallel simulations of 3-D cell colony dynamics," *Computing in Science & Engineering*, vol. 16, no. 5, pp. 86–95, 2014.

[9] M. Cytowski and Z. Szymanska, "Large-scale parallel simulations of 3D cell colony dynamics: the cellular environment," *Computing in Science and Engineering*, vol. 17, no. 5, Article ID 7106395, pp. 44–48, 2015.

[10] M. Cytowski and Z. Szymanska, "Enabling large scale individual-based modelling through high performance computing," in *Proceedings of the Workshop on Multiscale and Hybrid Modelling in Cell and Cell Population Biology (UPMC '15)*, article 00014, 5 pages, ITM Web of Conferences, Paris, France, March 2015.

[11] L. Szustak, K. Rojek, T. Olas, L. Kuczynski, K. Halbiniak, and P. Gepner, "Adaptation of MPDATA heterogeneous stencil computation to intel xeon phi coprocessor," *Scientific Programming*, vol. 2015, Article ID 642705, 14 pages, 2015.

[12] A. Lastovetsky, L. Szustak, and R. Wyrzykowski, "Model-based optimization of EULAG kernel on Intel Xeon Phi through load imbalancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 787–797, 2017.

[13] "On selected individual-based approaches to the dynamics in multicellular systems," in *Polymer and Cell Dynamics-Multiscale Modeling and Numerical Simulations*, D. Drasdo, W. Alt, M. Chaplain, M. Griebel, and J. Lenz, Eds., pp. 169–203, Birkhäuser, Basel, Switzerland, 2003.

[14] J. Galle, M. Loeffler, and D. Drasdo, "Modeling the effect of deregulated proliferation and apoptosis on the growth dynamics of epithelial cell populations in vitro," *Biophysical Journal*, vol. 88, no. 1, pp. 62–75, 2005.

[15] K. S. Nowiński and B. Borucki, "VisNow-a modular, extensible visual analysis platform," in *Proceedings of the 22nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '14)*, pp. 73–76, Pilsen, Czech Republic, 2014.

[16] J. Jeffers and J. Reinders, *Intel® Xeon Phi™ Coprocessor High-Performance Programming*, Morgan Kaufmann, 2013.

[17] J. Jeffers, J. Reinders, and A. Sodani, *Intel® Xeon Phi™ Processor High Performance Programming*, Morgan Kaufmann, Amsterdam, The Netherlands, 2nd edition, 2016.

[18] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang, "Scaling hypr's multigrid solvers to 100,000 cores," in *High-Performance Scientific Computing*, M. W. Berry, K. A. Gallivan, E. Gallopoulos et al., Eds., pp. 261–279, Springer, London, UK, 2012.

[19] E. G. Boman, Ü. V. Çatalyürek, C. Chevalier, and K. D. Devine, "The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: partitioning, ordering and coloring," *Scientific Programming*, vol. 20, no. 2, pp. 129–150, 2012.

[20] M. Mascagni and A. Srinivasan, "Algorithm 806: SPRNG: a scalable library for pseudorandom number generation," *ACM Transactions on Mathematical Software*, vol. 26, no. 3, pp. 436–461, 2000.

[21] J. Zielinski, "Discover, extend and modernize your current development approach for heterogeneous compute with standards based OFI/MPI/OpenMP programming methods on Intel® Xeon Phi™ architectures," in *Proceedings of the Intel®*

HPC Developer Conference, Salt Lake City, Utah, USA, 2016, http://www.intel.com/content/www/us/en/events/hpcdevcon/systems-track.html#discover.

[22] P. Van Liedekerke, M. M. Palm, N. Jagiella, and D. Drasdo, "Simulating tissue mechanics with agent-based models: concepts, perspectives and some novel results," *Computational Particle Mechanics*, vol. 2, no. 4, pp. 401–444, 2015.

[23] U. Wilensky and W. Rand, *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with Netlogo*, The MIT Press, 2015.