Supplementary material: Building multiscale models with PhysiBoSS, an agent-based modeling tool

Ruscone et al.

September 2, 2024

This supplementary material contains details about PhysiBoSS, in particular its implementation of a mapping system between PhysiCell and MaBoSS, and its implementation of time. It then includes details on how to install everything, and setup a new PhysiBoSS model. Next, it provides a step-by-step guide to reproduce the first three models presented in the article: the cell fate model upon TNF treatment ([1] and [2]), the cell cycle model (from [3]) and the T cell differentiation model upon dendritic cell stimuli (from [4]). All the models are available at the following GitHub page `https://github.com/PhysiBoSS/PhysiBoSS/tree/master/sample_projects_intracellular/boolean/tutorial/config`. Finally, it also includes an updated version of a cancer invasion model [5], now using the latest mapping functionalities. In the configuration folder of each model, the user can find the configuration files to reproduce the images of the main text and the supplementary materials. All the models have been developed and run using the PhysiCell graphical interface, PhysiCell Studio [6].

Contents

S1 Methods

S1.1 Agent-based modeling with PhysiCell

Agent-based modeling relies on a computational approach that uses autonomous, interacting software agents to study the behaviors of a system. An agent represents a single individual with its own state and behaviors that can react to other agents or the surrounding environment. Agent-based models allow for studying the emergence of complex population events from a simple set of agents' behaviors. In medical science, an agent can represent a cell that can interact with other cells or its microenvironment. With this approach, it is possible to simulate different biological scenarios, study collective cellular behaviors, and test hypotheses *in silico*. At present, many agent-based frameworks are available, with different characteristics to better answer different modeling needs [7]. In this context, the C++ PhysiCell framework uses a center-based approach, simulating mechanical and phenotypical cell dynamics, as well as the diffusion of substrates to represent cellular respiration, paracrine communication, and more [8]. PhysiCell enables the customization of the simulations through a general configuration XML file, with optional specifications of initial cell positions and cell rules in CSV files. More recently, it was extended with a *modeling grammar* that connects signals (e.g., diffusing chemical factors) with changes in cell behaviors, to help users straightforwardly model the stimuli perceived by an agent and its behavioral reactions [9]. PhysiCell includes dictionaries of available signals (components or contexts of the environment) and behaviors (cell functionalities controlled by a parameter) for use in PhysiBoSS models, and PhysiCell Studio can use these pre-populated dictionaries to graphically construct model rules.

S1.2 Logical modeling with MaBoSS

Logical modeling provides an efficient way to study and represent complex behavioral patterns in biology. This method involves representing biological entities, such as genes, proteins, or full pathways, as nodes within a network. Using a Boolean approach, each node is a variable of the model that can take two values, 0 for absent or inactive and 1 for present or active, and the update of these variables is monitored by logical rules that link all the inputs of a node with the logical connectors OR, AND, and NOT. This type of models can be used to explore patients' responses by simulating various initial conditions and accounting for mutations observed in patients by forcing the values of the corresponding variables in the model. MaBoSS is a C++ software package for simulating Boolean models using continuous time Markov processes [10, 11]. It applies an asynchronous update scheme, which allows the description of het-

erogeneous responses. By associating transition rates to each variable, for both activation and inactivation, it generates continuous trajectories with a notion of physical time. MaBoSS uses two files for describing the model: the BND file which contains the information about the Boolean network, and the CFG file which contains the simulation settings. MaBoSS computes time trajectories of Boolean models. It either simulates until a time long enough to reach an asymptotic solution: in this condition, the stable states, or attractors of the model, are estimated. But it can also simulate until a specific time when the system will still be in a transient phase, in which case we will be able to capture transient effects.

S1.3 PhysiBoSS framework

PhysiBoSS is an add-on of PhysiCell that integrates a MaBoSS engine inside each agent. This approach adds a new layer of description of the cell, with a specific Boolean model that represents the cell's intracellular signaling dynamics. The Boolean network can be the same for all the cells or separate networks can be assigned to each cell type. At each simulation step, the agent (cell) can collect different stimuli that modify the activity of some specific nodes of the network (input nodes). Next, the MaBoSS engine computes the model trajectory that can cause the switch of the so-called phenotypic nodes (or output nodes). Those nodes can then trigger some specific cell actions (motility, secretion, uptake, death, etc.). PhysiBoSS uses as input the same configuration files of PhysiCell, and the BND and CFG MaBoSS input files.

S1.4 Mapping agent-based to intracellular models

PhysiCell provides a dictionary of signals and a dictionary for behaviors, aimed at giving better accessibility to all the signals perceived by each agent and all possible behaviors that an agent can express. PhysiBoSS uses these data structures to simplify the connection between PhysiCell and MaBoSS, giving access to the PhysiCell/MaBoSS mapping through the configuration file and so, drastically diminishing the amount of C++ code necessary to develop a model. Mapping can be of two types: (1) input mapping, which links a PhysiCell signal to a MaBoSS (input) node by using activation thresholds, or (2) output mapping, which links a MaBoSS (output) node to a PhysiCell behavior by using values representing the Boolean state.

S1.4.1 Input mapping

An input mapping will modify the value of a Boolean input node according to the value of a PhysiCell signal, which means we will have to convert a real number value into a Boolean value. To do this, we will use a simple construct based on thresholds. The simplest case would be the following :

$$boolean = \begin{cases} 1, & \text{if } signal \geq threshold \\ 0, & \text{otherwise} \end{cases}$$

However, we want to be able to describe hysteresis, where a Boolean value will not only depend on the value of the signal but also its history of activation. To do this, we add another threshold value for the inactivation of the Boolean value :

$$boolean = \begin{cases} 1, & \text{if } signal \geq threshold \text{ and } boolean = 0 \\ 0, & \text{if } signal \leq inact\_threshold \text{ and } boolean = 1 \\ boolean, & otherwise \end{cases}$$

Note that using this construct, we need to define an initial value for our input node.

To avoid noisy values in a signal, we also implemented a notion of smoothing of the signal for controlling the (in)activation of the Boolean input node, which is controlled by the `smoothing` parameter. This mechanism will compute an average value for the signal on a time window $[t-(intracellular\_dt \times smoothing); t]$. For $smoothing = 0$ (the default value), this mechanism is not active since the time window only contains t.

Finally, since we may want to have an inverse relationship between a signal and a behavior, we added a parameter called `action` which can take two values: `activation` or `inhibition`. The behavior described above concerns an activation, and for an inhibition, the opposite happens :

$$boolean = \begin{cases} 0, & \text{if } signal \geq threshold \text{ and } boolean = 1 \\ 1, & \text{if } signal \leq inact\_threshold \text{ and } boolean = 0 \\ boolean, & otherwise \end{cases}$$

S1.4.2 Output mapping

Contrary to input mapping, an output mapping links a Boolean output node to a PhysiCell behavior, meaning that an agent will change its behavior based on its intracellular state. More practically, this means converting a Boolean value into a real number value. To achieve this, we will associate a real number to each Boolean state :

$$behavior = \begin{cases} value, & \text{if } boolean = true \\ base\_value, & \text{if } boolean = false \end{cases}$$

Note that when no `base_value` is provided, it takes the default value which is 0.

Similarly to what has been done for input mappings, we also implemented a notion of smoothing, which is the following: we compute a probability for the Boolean value to be true between $[t - (intracellular\_dt \times smoothing; t]$, based on the following formula :

$$p(t+1) = \frac{p(t) * smoothing + boolean}{smoothing + 1}$$

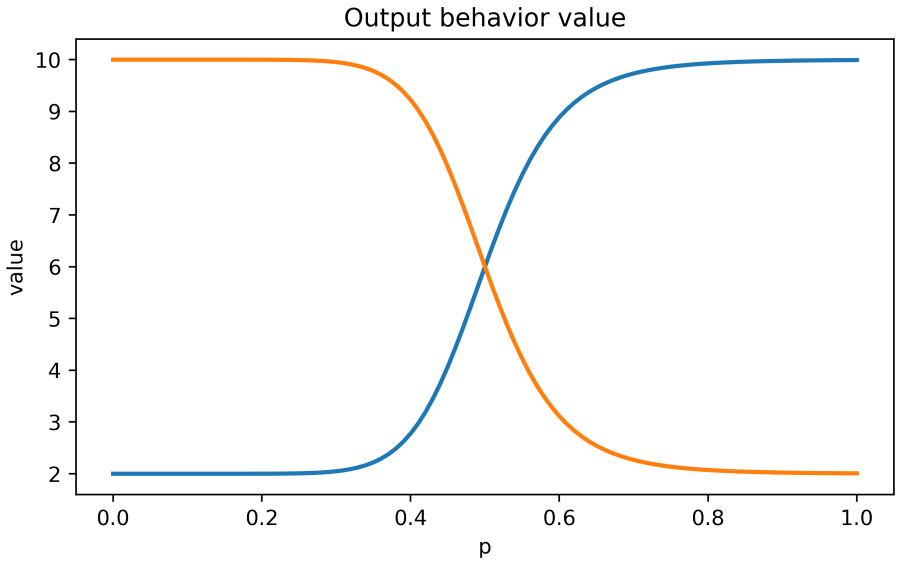


Figure S1: Sigmoid function used to describe the value of the behavior for an activation (blue) and for an inhibition (orange), as a function of the smoothed probability (p) of the Boolean output node, here with $base\_value = 2$ and $value = 10$

We then use this probability as input for a Hill function as seen in Figure S1, which describes a sigmoid with, in the case of an activation (blue curve), $behavior = base\_value$ if the probability is equal to 0, and $behavior = value$ if the probability is equal to 1. In the case of an inhibition (orange curve), $behavior = value$ if the probability is equal to 0, and $behavior = base\_value$ if the probability is equal to 1.

S1.5 Time synchronisation

PhysiCell updates its internal mechanisms regularly, depending on the time scale of the process described: diffusion of molecules in the environment is governed by a short time scale, so its update has to be performed very often (usually every second), while cell cycle progression is governed by a large time scale so its update is performed less frequently (usually every few minutes). A similar mechanism is used for PhysiBoSS, where the status of the MaBoSS model will be updated at every time step.This update for each agent separately, using a single Markov chain simulation. Since this is model-dependent, we leave it to the user to define the appropriate time step.

Now let us suppose that our PhysiBoSS model is updating every 10 minutes, meaning that every 10 minutes, we update the inputs of the MaBoSS model,

simulate it, and then obtain its output nodes (phenotypes). The follow-up question is for how long should we simulate our MaBoSS model. A MaBoSS model has its own time unit since it contains kinetic rates for (in)activation, which is not always the same as the PhysiCell unit (usually minutes) and we might have to adapt the simulation duration to the PhysiCell unit. This is done using the scaling parameter. Let us suppose that the MaBoSS model's unit is hours. Then we need to set our scaling parameter to 60 so that every 10 minutes, PhysiBoSS will run the MaBoSS model for $intracellular\_dt/scaling = 10/60 = 1/6$ hours. Another way to see this scaling parameter is that if one wants to run a MaBoSS simulation for 20 (MaBoSS time unit), the following equation needs to be computed: $scaling = intracellular\_dt/maboss\_simulation\_time$.

S1.6 Time stochasticity

By default, an intracellular update is governed by the value of `intracellular_dt`, a fixed number. This will cause all the cells to stay completely synchronized in their updates. Since updates happen for each agent independently, we can avoid this phenomenon. To this end, we introduced a parameter called `time_stochasticity`, which will allow the description of the update time as a log-normal distribution with $\mu = \log(intracellular\_dt)$ and $\tau = time\_stochasticity$. The reason to choose such a distribution is to bound the update time in $[0, \infty]$ and to have an equal probability for a cell update to take half as long or twice as long, as seen in Figure S2.

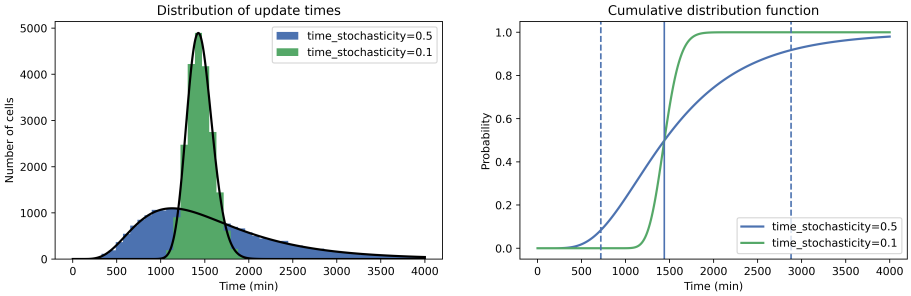


Figure S2: Left: Distribution of update times for `intracellular_dt`=1440min, with `time_stochasticity`=0.5 (blue) and `time_stochasticity`=0.1 (green). Black lines show the log-normal distribution fitted to the values. Right : Cumulative distribution function, for `intracellular_dt`=1440min, with `time_stochasticity`=0.5 (blue) and `time_stochasticity`=0.1 (green). The plain vertical line shows the median value, corresponding to the defined `intracellular_dt`=1440min. The dashed vertical lines correspond to half and double `intracellular_dt`.

S2 Setting up a PhysiBoSS project

S2.1 Installation of PhysiBoSS and PhysiCell-Studio

PhysiBoSS and PhysiCell-Studio can be installed using the following commands:

```
git clone https://github.com/PhysiBoSS/PhysiBoSS.git
git clone https://github.com/PhysiCell-Tools/PhysiCell-Studio.git
```

Two folders should be created in a working directory: PhysiBoSS and PhysiCell-Studio.

S2.2 Installation of the environment for PhysiCell-Studio

To ensure the PhysiCell-Studio work properly, an environment that contains all the dependencies of PhysiCell-Studio should be created and activated. This can be done through, for example, the anaconda package (`https://www.anaconda.com/`). To install the environment provided with PhysiCell-Studio, navigate into the folder from a terminal and use the following commands:

```
cd PhysiCell-Studio
conda env create -f environment.yml
cd ..
```

This command will start the creation of the environment and the download of the dependencies. To activate the environment simply execute:

```
conda activate studio
```

Once the set-up of PhysiCell-Studio is done, we can move to the PhysiBoSS folder to set up a model.

```
cd PhysiBoSS
```

S2.3 Using packaged models

We are providing, in the latest release, packaged models with pre-compiled binaries for Linux, Mac, and Windows to run the different models described in this article (and that could be compatible with many more PhysiCell models), to simplify the steps to reproduce our work. These binaries are still highly experimental and might not work on some machines, in which case you need to follow the instructions in the following section to load the model and compile yourself PhysiBoSS.

To download the packaged model for the PhysiBoSS template, and start building a PhysiBoSS model, you can run the following Python command from the PhysiCell root directory :

```
python beta/download_binary.py template_BM
```

This will recognize your operating system, download the appropriate binary, and put it in your root directory under the name `project`. It will also download associated config files and source files in case you want to modify the code

and/or recompile the binary. The template_BM model is the starting point for this tutorial. However, if you want to directly run the examples of this tutorial, you can also use this script to download them.

To download the packaged model for the three main examples, you can run :

```
python beta/download_binary.py physiboss-tutorial
```

If you want to run the cancer invasion model, you can run the following command :

```
python beta/download_binary.py physiboss-tutorial-invasion
```

In Linux, these binaries are compiled for running with GLIBC 2.32 to 2.34 (Ubuntu 22 LTS). A system with older versions of GLIBC will produce an error:

```
libc.so.6: version ‘GLIBC_2.32‘ not found.
```

In this case, please compile locally your project using the following section.

In recent MacOS versions, these binaries are by default prevented from running. After a first try to run it, you need to explicitly allow them to run in the Settings > Privacy and security options, where a notification will appear and an option will be available to authorize them. In some cases, binaries will fail to run, without any specific error message. In this case, please compile locally your project using the following section.

S2.4 Loading a model into the directory structure

PhysiCell models provided by the development team are archived in the folder `sample_projects` and `sample_projects_intracellular`. To use them, we first need to load them into the proper directory structure by using the command :

```
make template_BM
```

Here we are loading the `template_BM` project, which is the template for PhysiBoSS projects. This template can be modified to build new PhysiBoSS models. Once loaded, the directory structure should look like the one in S3, except for the `project` binary that we will explain in the next two sections.

S2.5 Compilation of the PhysiBoSS project

We can set a model starting from the template_BM provided by PhysiCell. This template already includes the function necessary to incorporate a Boolean network in each cell type. In a terminal, we use the following command to compile the template_BM project:

```
make -j
```

If this is the first time compiling a project with PhysiBoSS, it will automatically download the required library to run MaBoSS inside PhysiCell.

It is important to note that the template_BM model can only run MaBoSS models up to 256 nodes. To increase this value, you can modify the value of the `MABOSS_MAX_NODES` variable in the Makefile (line #17).

By default, the compilation of the template model will produce a binary called `project`. If you want to change this, you need to edit the Makefile and change the value of the variable `PROGRAM_NAME` (line #2).

Once the project is compiled, it is necessary to copy the .bnd and .cfg files in the folder boolean_network, located inside the config folder. The structure should look like in Figure S3.

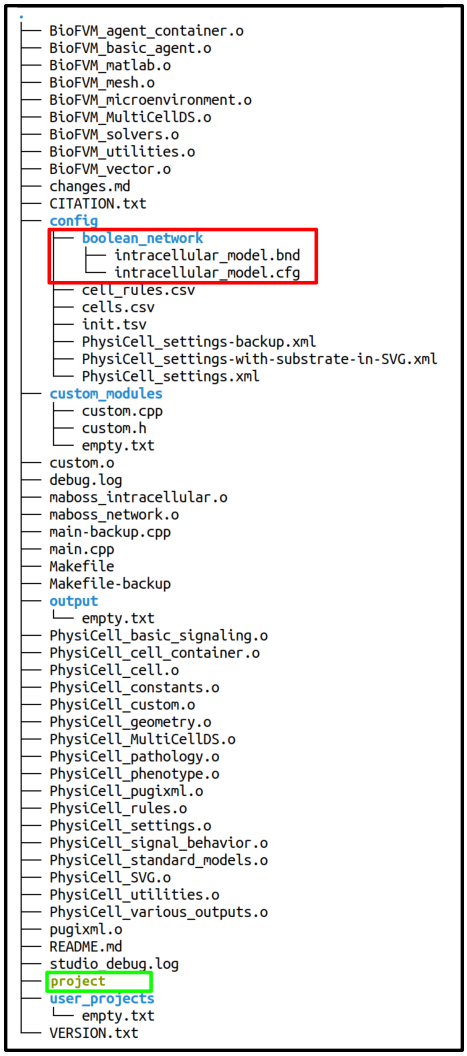


Figure S3: Tree structure of the PhysiBoSS folder once compiled the template. The rectangle in red indicates the position of the intracellular files, while in green the name of the executable.

If everything is set correctly, we are ready to open PhysiCell Studio and

start building the model.

S2.5.1 Troubleshooting

Linux Some Linux installations will come with only minimal packages, lacking the dependencies of PhysiBoSS. On Ubuntu systems, you can install these dependencies using :

```
apt install gcc g++ make flex bison python3
```

On some Linux installations, the available GCC version will be older than the one MaBoSS libraries were built with, which might lead to this type of error message :

```
undefined reference to ‘std::__throw_bad_array_new_length()‘
```

In this case, you need first to recompile the MaBoSS library using the locally installed GCC :

```
make Compile_MaBoSS
make
```

MacOSX MacOSX comes by default with a version of Clang C++ compiler which is not compatible with OpenMP, the parallel computing library used by PhysiCell. This will lead to this error message :

```
error: unsupported option ’-fopenmp’
```

To solve this, you need to install the GCC C++ compiler using Brew, and specify that you want to use this compiler when compiling PhysiCell :

```
brew install gcc@11
make PHYSICELL_CPP=g++-11
```

In some cases, there will be an incompatibility between the locally installed GCC version and the GCC version used to compile MaBoSS library. This will result in this type of error message :

```
ld: symbol(s) not found for architecture x86_64
```

In this case, you need first to recompile MaBoSS library using the locally installed GCC:

```
make Compile_MaBoSS PHYSICELL_CPP=g++-11
make PHYSICELL_CPP=g++-11
```

This supposes that you have GCC 11 installed on your machine. If it is another version, replace `g++-11` by `g++-<version>`.

Windows PhysiBoSS can be compiled on Windows using MinGW-w64 packages. To install them, you first need to download and install MSYS2, available at `https://msys2.org`. Then, after installing MSYS2, you need to open an MSYS2 Command Prompt and run the following command :

```
pacman -S mingw-w64-x86_64-binutils mingw-w64-x86_64-gcc mingw-w64-x86_64-headers
    -git mingw-w64-x86_64-gcc-libs mingw-w64-x86_64-libwinpthread-git mingw-w64-
    x86_64-winpthreads-git mingw-w64-x86_64-lapack mingw-w64-x86_64-openblas
    mingw-w64-x86_64-libxml2 mingw-w64-x86_64-bzip2 mingw-w64-x86_64-python git
    make flex bison
```

This will install the necessary dependencies. You can then use the commands mentioned in the compilation section in a MinGW-w64 Command Prompt.

A more detailed installation tutorial for Windows is available at: https://github.com/physicell-training/ws2023/blob/main/setup/PhysiCell/ws2023_Windows_setup.pdf.

S2.6 Using PhysiBoSS with PhysiCell Studio

PhysiCell Studio supports the development of models using PhysiBoSS. It provides the user with all the necessary tools to set an intracellular model and connect it to the agent's signals/behaviors. We can open PhysiCell Studio from the terminal using the following command:

```
python ../PhysiCell-Studio/bin/studio.py \
       -e project -c config/PhysiCell_settings.xml
```

Not that here, `project` refers to the name of the compiled executable, and `config/PhyiCell_settings.xml` refers to the PhysiCell XML settings files describing the model. These values can be changed depending on the specific model.

If no errors occur, the welcome page of PhysiCell should show up as in Figure S4.

Most of the tabs in PhysiCell Studio offer functionalities for editing the model (Config Basics, MicroEnvironment, Cell Types, User Params, Rules, ICs). The last two tabs are dedicated to running the model (Figure S5) and visualizing the results (see next section).

S2.7 Visualizing results in PhysiCell Studio

PhysiCell Studio allows the visualization of the simulation results generated by PhysiCell in a simple, intuitive graphical interface. It allows the visualization of any time points, play the simulation as a video, and choose the element of your model you want to emphasize.

By default, its visualization only shows cells, colored by cell type. This is the default mode for the SVG files generated by PhysiCell (Figure S6A).
A useful additional visualization is the plotting of the substrate concentration in space (Figure S6B). This allows the user to see the gradient and is extremely practical when defining the parameters of diffusion/decay of your substrate. If you want to look deeper into the state of each individual cell, you can select the .mat representation, which allows you to look for many different quantities in your cell and choose them to color the cells. In Figure S6C you can see an example of this, with the cells being colored according to their current phase

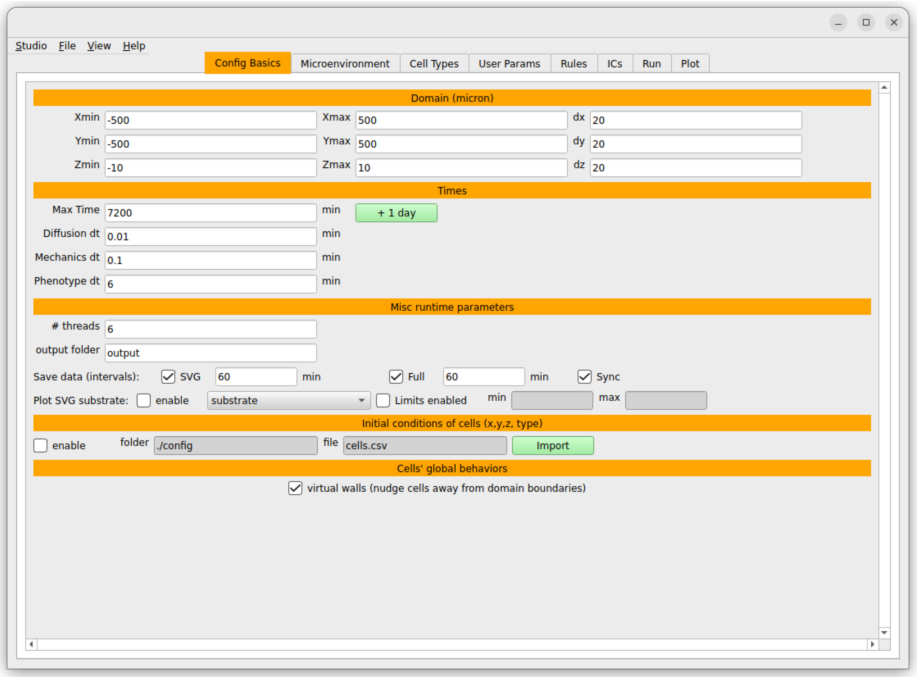


Figure S4: Welcome page of PhysiCell Studio

(live, apoptotic, or necrotic).

Moreover, with PhysiBoSS we added another visualization model, which uses the state files PhysiBoSS is generating, saving the state of the Boolean network for each cell at each time point. With this, you can choose any cell type, and within its network any Boolean node, and represent its activity in each cell by coloring it in green/red (Figure S6D). Note that only the nodes defined as output (non-internal) in the MaBoSS model CFG file will be included in the list of nodes that can be used.

Finally, PhysiCell Studio allows us to print the population size in time, according to different aspects of the model (cell types, current phase, etc.) (See Figure S7A). With PhysiBoSS we also included this possibility, by plotting the trajectory of the population size for each Boolean state (Figure S7B). Note that with too many nodes declared as output nodes, the state space will explode and this option will not be usable.

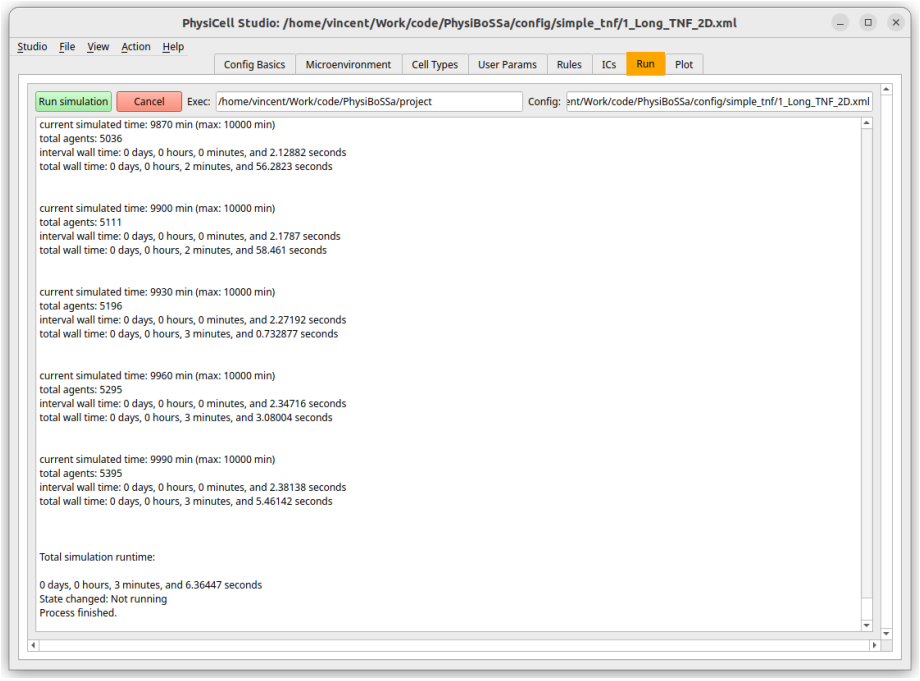


Figure S5: Running a model in PhysiCell Studio. After choosing the proper binary and XML settings file to execute, you can run the model by pressing the top left button. Logs of the simulation will appear in this tab.

S2.8 Adding a module for timed treatments

To model treatments at specific timing, as it will be used in the cell fate example, we need to add a new module to PhysiCell, which will provide the source code for implementing this functionality. Future releases of PhysiCell are planned to include this feature using the PhysiPKPD[12] addon, but for this version, we created our own implementation. The code is available in Figure S8. To add it to our project, you need to copy this function to custom.cpp, and to add its declaration to custom.h. Finally, you need to add the call to `treatment_function` in the main loop of PhysiCell in main.cpp, just before the diffusion update (Line #221). For simplicity, we already added this module to the `template_BM` project.

S3 Cell fate model upon TNF treatment

The model is an update and revision of the one showcased in [2]. The aim is to reproduce a growing tumor and to test the effect of different TNF treatments, that vary in length and duration. Finally, the model explores the impact of

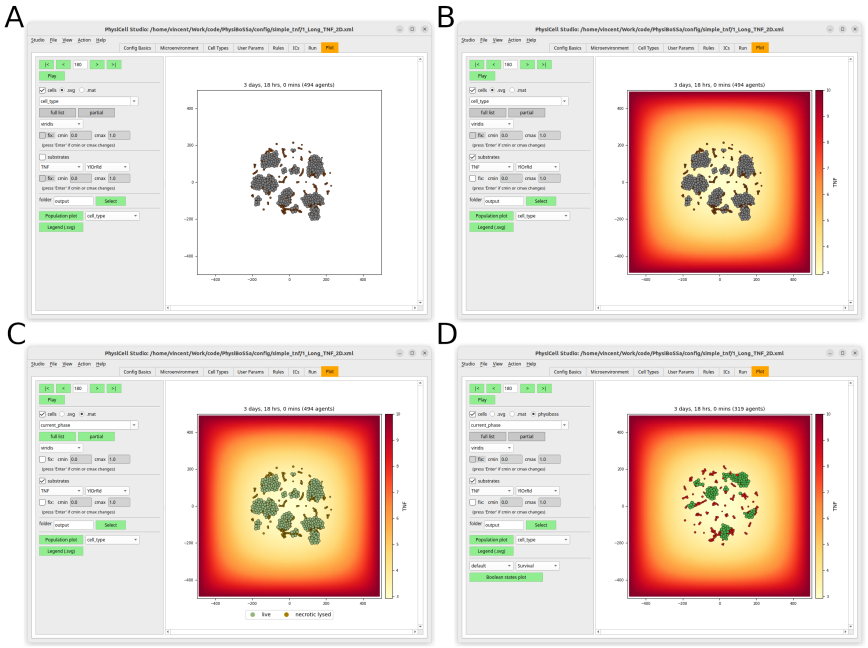


Figure S6: Visualizing a simulation result in PhysiCell Studio. In the *Plot* tab, you can look at an existing simulation folder, and visualize most components of the model. A: Default visualization, using the SVG images generated by PhysiCell. B: Additional visualization of the substrate concentrations. C: Visualizing the content of the .mat files, containing more information about the PhysiCell model. D: Visualizing the content of the PhysiBoSS state files, by coloring a cell according to the activity of a particular node.

mutants on the growth of the cell population and its resistance to TNF.

S3.1 Intracellular model

This model describes the pathways involved in cell fate decisions in response to death receptor signals. It includes the NF-κB pro-survival pathway, RIP1 necrosis pathway, and the apoptosis pathway (Figure S9).

In this article, we will focus on the effect of one of the death receptor signals: TNF. In the absence of TNF treatment cells remain uncommitted to a cell fate, in their naive state (represented by the <nil> state, Figure S10A). Upon stimulation with TNF, most cells commit to one of the cell fates: the vast majority (95%) commit to apoptosis, 3% to necrosis (NonACD), and 2% commit to survival (Figure S10B).

A very interesting aspect of this model is its response to pulsative treatments

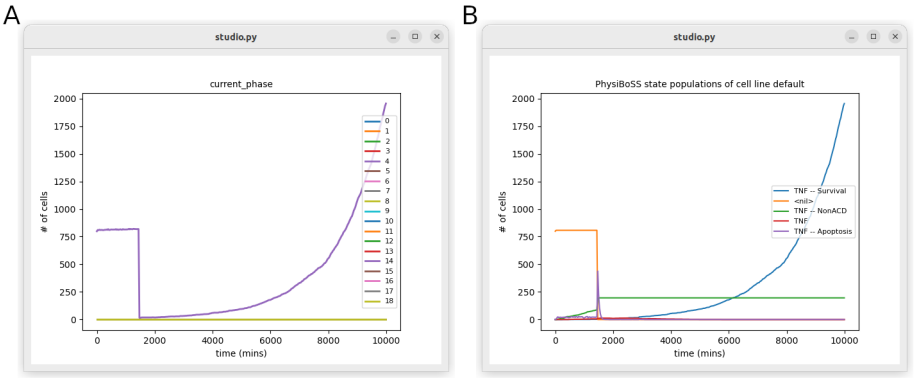


Figure S7: Visualizing the time-series of population size by current phases (A) and by Boolean state (B).

of TNF. While the death cell fate remains unchanged by a pause in the treatment, the cells which committed to survival (and are thus resistant to the TNF treatment) go back to their naive state. After a second round of treatment, these cells will commit to the same proportions of cell fates: mostly apoptosis, with a small percentage of necrosis and survival. What this means is that with repeated treatment, this population of resistant cells can be eliminated, as can be seen in Figure S9B.

A detailed analysis of this model is available in the Jupyter notebook CellFate_Analysis.ipynb, provided in the supplementary materials.

S3.2 Creating initial positions

To create an initial cell distribution corresponding to a spheroid tumor, we go to the *ICs* tab, and choose an annulus/disk with hexagonal fill, with parameters (x0, y0) = (0,0), R1 = 0, R2 = 250. This will create 800 cells disposed on a disk. We can then save it to the config/simple_tnf folder as cells.csv. This step is represented in Figure S12.

Then, we go back to the *Config Basics* tab to enable the initial condition of cells and select the cells.csv file we just created in the config/simple_tnf folder.

S3.3 Adding TNF substrate

The model revolves around TNF a cytokine that influences many cellular processes. In our model, TNF is represented as a substrate that diffuses from the border of the domain of the simulation. The TNF is not secreted or uptaken by the cell but will be used as a signal to trigger the intracellular model cascade. To set the substrate, in the *Microenvironment* tab, we add a new substrate, rename it TNF, and assign a diffusion coefficient of 1200 micron^2/min and a

```
void treatment_function ()
{
  if (PhysiCell::parameters.bools.find_index("treatment") != -1)
  {
    int treatment_substrate_index = BioFVM::microenvironment.find_density_index(
    PhysiCell::parameters.strings("treatment_substrate"));

    if (PhysiCell::parameters.bools("treatment")){

      if ((((int)PhysiCell::PhysiCell_globals.current_time) % PhysiCell::
    parameters.ints("treatment_period")) == 0
        && !BioFVM::microenvironment.get_substrate_dirichlet_activation(
    treatment_substrate_index)
      )
      { BioFVM::microenvironment.set_substrate_dirichlet_activation(
    treatment_substrate_index, true); }

      if ((((int)PhysiCell::PhysiCell_globals.current_time) % PhysiCell::
    parameters.ints("treatment_period")) == PhysiCell::parameters.ints("
    treatment_duration")
        && BioFVM::microenvironment.get_substrate_dirichlet_activation(
    treatment_substrate_index)
      )
      { BioFVM::microenvironment.set_substrate_dirichlet_activation(
    treatment_substrate_index, false); }

    } else if (BioFVM::microenvironment.get_substrate_dirichlet_activation(
    treatment_substrate_index) )
        { BioFVM::microenvironment.set_substrate_dirichlet_activation(
    treatment_substrate_index, false); }
  }
}
```

Figure S8: Source code of the drug treatment function

decay rate of 0.0275 1/min. These values where chosen to have a TNF gradient around the tumor, with a value of at least 1 mmHg reaching at least the center of the tumor.

We set Dirichlet boundary conditions to 10 mmHG too, even though the TNF can be integrated through a custom function designed for this model and controlled by some user parameter in the *User Params* tab. These settings can be visualized in Figure S13. A settings file corresponding to this step is available in config/simple_tnf/0_Initial.xml.

S3.4 Configuring cell definition

In the *Cell Types* tab, we customize the cell to react to the TNF stimuli. We can use the default cell types provided by PhysiCell, making sure that the selected Cycle model is the "live cells" with a transition rate of 0.0006 1/min and both Apoptosis and Necrosis rates (in the *Death* sub-tab) are set to 0.

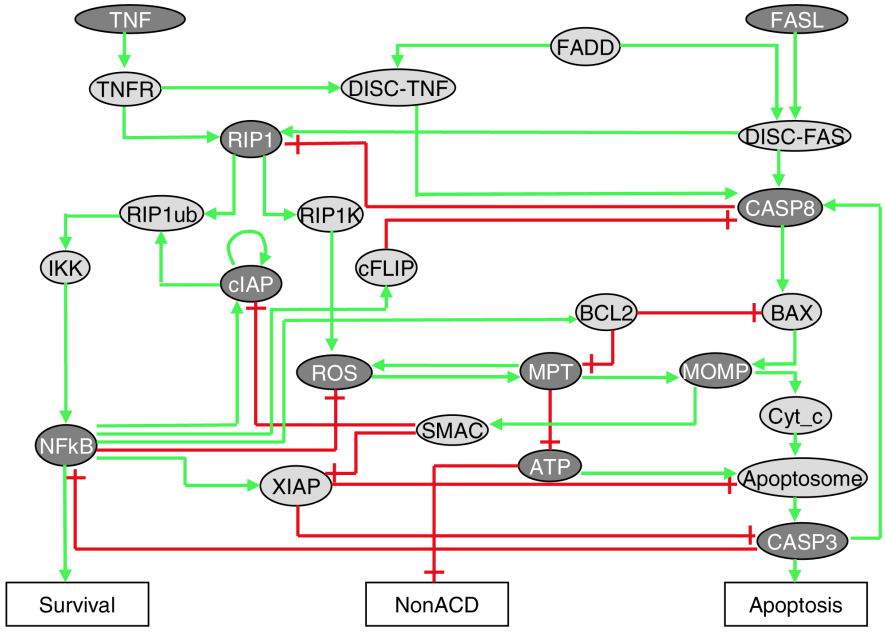


Figure S9: Interaction graph of the cell fate model from Calzone et al. [1]

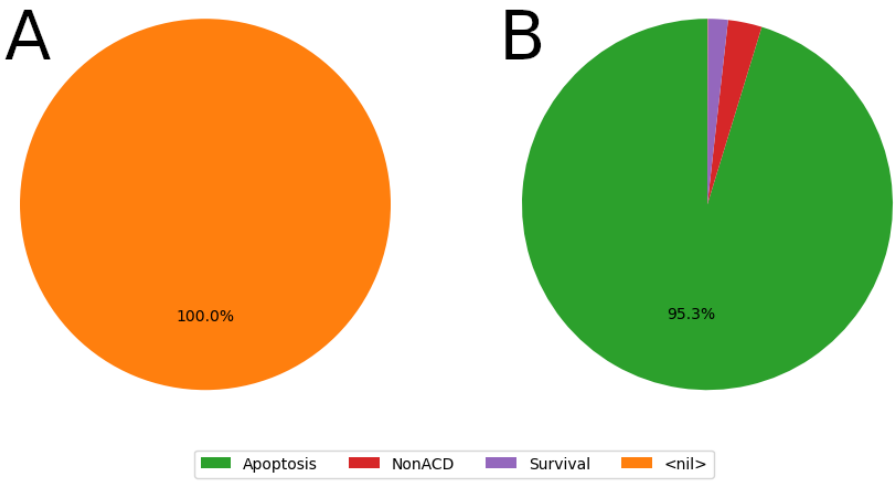


Figure S10: Final distribution of states of the cell fate model, with (A) and without (B) TNF treatment.

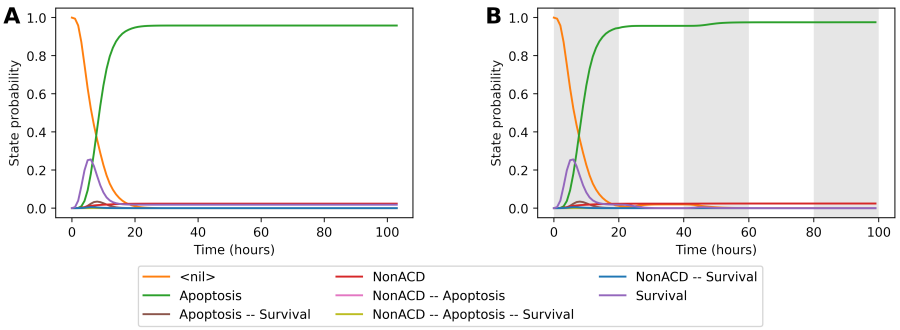


Figure S11: Simulation trace of the Boolean cell fate model after TNF treatment following the probability for the 3 phenotypes to activate. A) Long TNF treatment will reach a steady state after 24 hours, where most cells will activate either apoptosis or necrosis, and around 2% of the cells will survive the treatment. B) Pulsative TNF treatment (grey areas representing the TNF treatment).

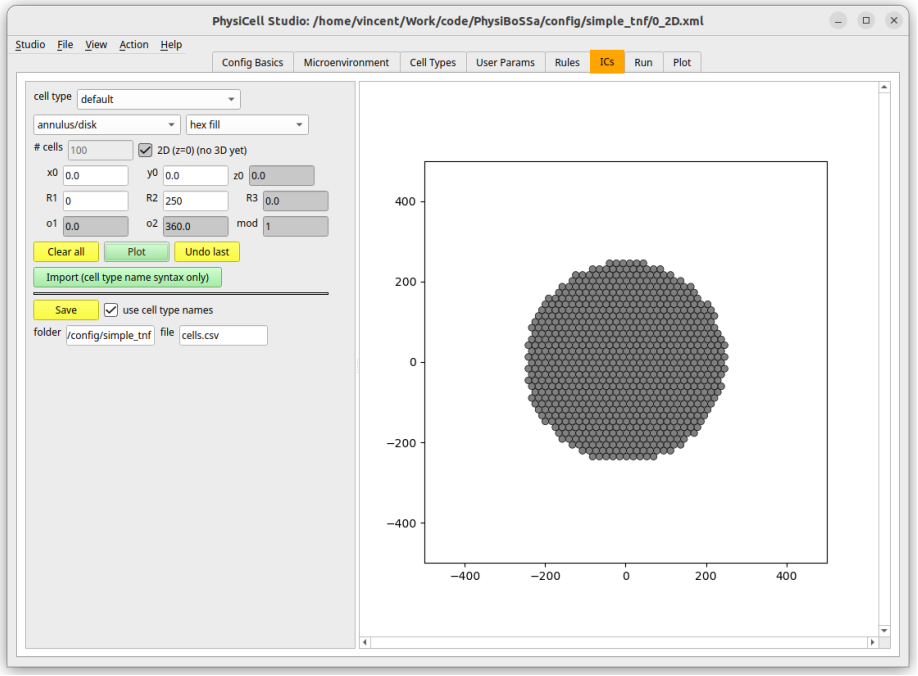


Figure S12: Settings of the initial position of the cells for the Cell fate model.

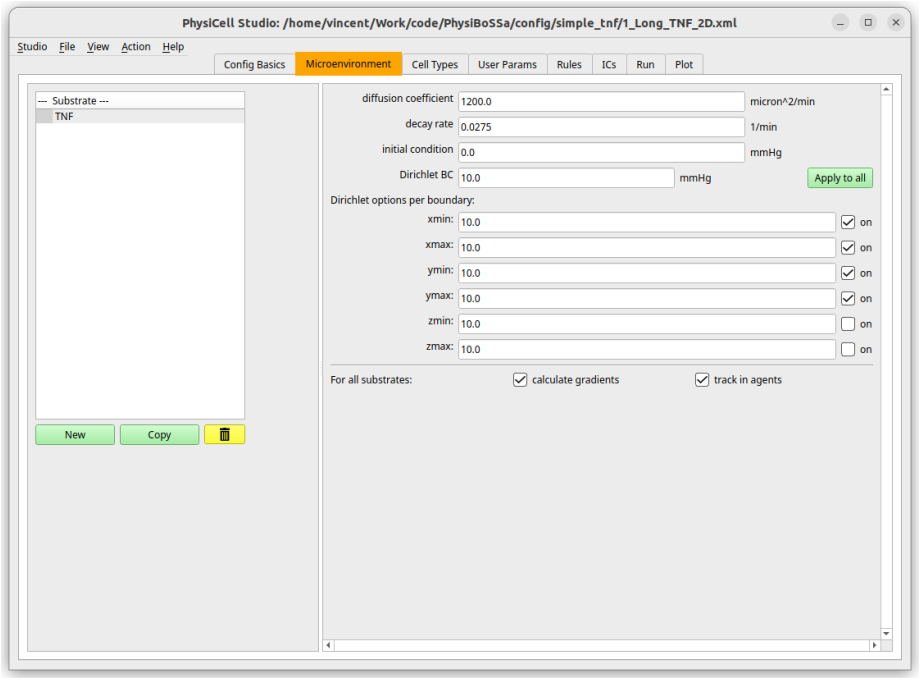


Figure S13: Creating the TNF substrate in the *Microenvironment* tab for the Cell fate model.

The main intervention on the agent is performed in the *Intracellular* subtab, where we coupled the signal and behaviors with the intracellular MaBoSS model. First, we select the BND and CFG files that contain the specification of the Boolean model available at the following link: `https://github.com/PhysiBoSS/PhysiBoSS/tree/master/sample_projects_intracellular/boolean/tutorial/config/simple_tnf/boolean_network`. The next step consists in setting the intracellular_dt, which is the period at which the Boolean model is updated. We set it such that the model has enough time to reach a steady state, which is around 50 seconds. Then, to connect the intracellular model to the rest of the PhysiCell simulation, we create one input mapping and two output mapping:

- A new input is created in the mapping, which links together the TNF substrate (the "TNF" Signal) to the node "tnf" of the Boolean network. We set it to activate the node if the TNF concentration around the cell is larger than 1.
- Next, in the output, the node Apoptosis is linked to the rate of the Apoptosis model in PhysiCell. We set it to a very high value (10^6) so that

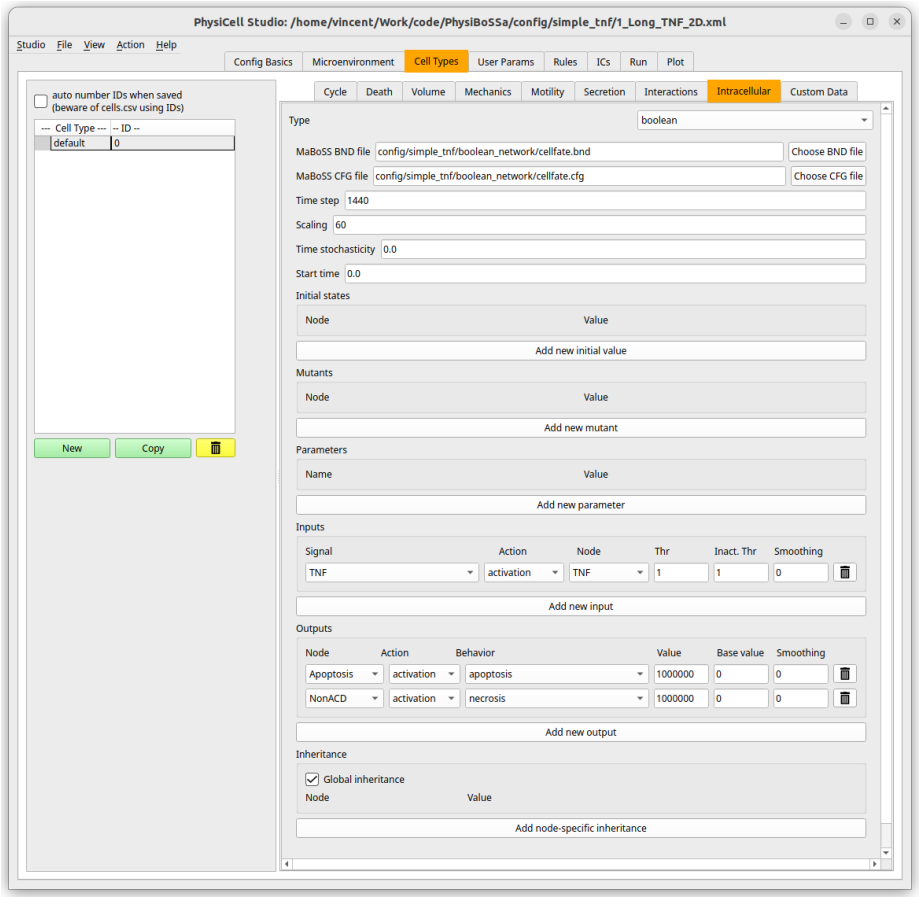


Figure S14: Setting up the intracellular model of the default cell type for the Cell fate model.

apoptosis will be immediately triggered when the Apoptosis node turns active.

- The second output links the NonACD (Non Apoptotic Cell Death) to the rate of the Necrosis model of PhysiCell, in the same way we set up Apoptosis.

These settings can be visualized in Figure S14. With this mapping defined, we can already simulate our model, and we see that TNF triggers mainly apoptosis and necrosis, with a small proportion of the cells surviving and becoming resistant to the treatment.

S3.5 Create long and short TNF treatments simulation

Before describing the timings of the treatments, we select, in the *Config Basics* tab, the duration of the simulation, by setting it to 10000 (min) in the Max Time field. This corresponds to nearly 7 days.

Four user parameters encode the details of the treatment schedule. Let us see how to first create a simple long TNF treatment and later a pulsative treatment.

To activate the treatment, the parameter `treatment` needs to be set to True, and the proper substrate (here TNF) needs to be selected in the `treatment_substrate` parameter.

Then, we have two parameters controlling the timing of the treatment: `treatment_period` and `treatment_duration`. For a long treatment, we need to set both of them for a duration longer than the simulation duration, ensuring that the treatment will not be turned off during the simulation. Here, we choose both the treatment duration and period to be 10500 minutes so that it will just do one long treatment, longer than the 10000 minutes of the simulation (Figure S15).

In the first version of the treatment, we use `time_stochasticity`=0, which is the default value. This has the effect of synchronizing all the cells: the intracellular updates for each of them will happen simultaneously. The trajectory showing the number of cells with specific phenotypes can be seen in Figure S16A, where we see that at $t = 1440$, the first update after TNF is activated and all the cells notify it at the same time, leading to a very abrupt death of the population. This synchronicity seems very artificial, and to a biologist, it would just look wrong: even synchronized cells lose synchronicity very quickly. To reproduce better what happens in biology, we added an option to have stochasticity in the update time of our cell. The implementation of this time stochasticity in described in Section S1.6. In another simulation, we chose `time_stochasticity`= 0.5, which leads to a more gradual death of the population upon TNF activation as seen in Figure S16B. The settings files corresponding to the model at this stage are available, with and without the time_stochasticity, in config/simple_tnf/1_Long_TNF.xml and config/simple_tnf/1_Long_TNF_stochastic_time.xml, respectively.

S3.6 Creating necrotic core

For the pulsative simulation, we will have to define the period of the treatments and their duration. These two settings will be very important for the efficiency of the treatment: the treatment duration needs to be longer than the update time of the intracellular model, to be properly taken into account by the mapping, and shouldn't be too long, as it would lead to a large resistant cell population. After trial and error, we chose a treatment period of 3440 minutes, with a duration of 2000 minutes, as we found that with these settings we obtained a decreasing tumor size. An XML settings file corresponding can be found in config/simple_tnf/2_Short_TNF_2D.xml.

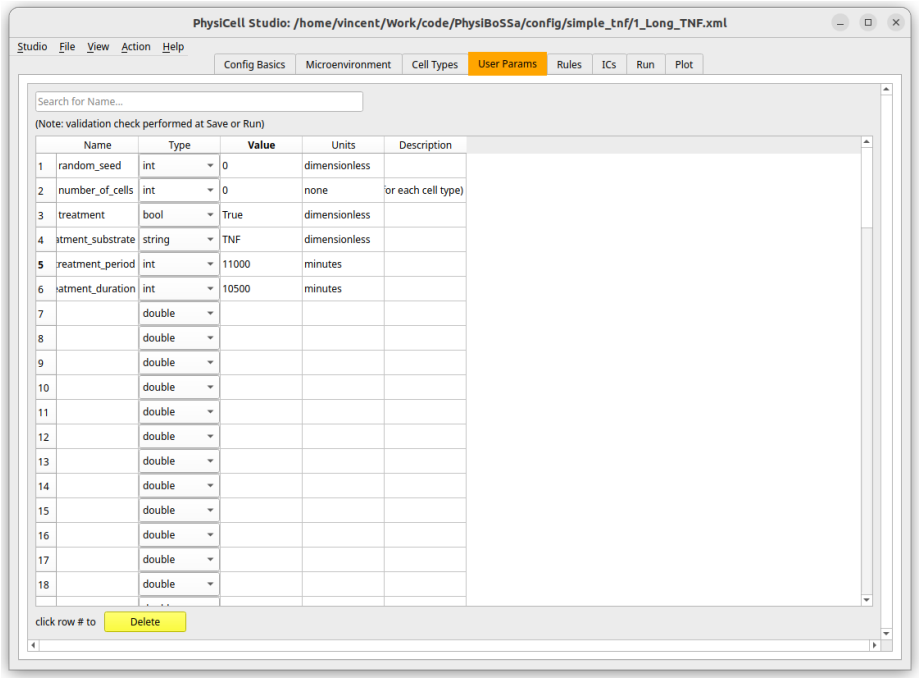


Figure S15: Setting the timings of the TNF treatment as user parameters

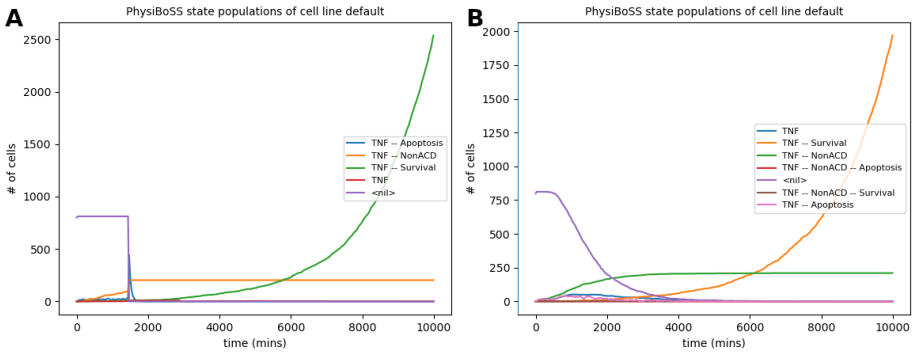


Figure S16: Trajectory of the simulation of a long TNF treatment, with $time\_stochasticity = 0$ (A) and $time\_stochasticity = 1$ (B)

Here we want to reproduce a necrotic core inside the tumor. This behavior is caused by a lack of access to Oxygen for the cells in the center of the tumor, triggering necrosis. To do this, we need to add Oxygen supply to our simulation, then add it to our Boolean network and make it trigger necrosis, and finally

update the mapping.

First, to add Oxygen to the environment, we go to the *Microenvironment* tab, and we add a new substrate called Oxygen, with diffusion=1000 micron^2/min, decay rate=0.01 1/min, initial and all Dirichlet conditions = 38. Dirichlet conditions for xmin, xmax, ymin and ymax must be active.

```
node OXYGEN
{
  rate_up = 0.0;
  rate_down = 0.0;
}

node NonACD
{
  logic = !OXYGEN | !ATP;
  rate_up = @logic ? 1.0 : 0.0;
  rate_down = @logic ? 0.0 : 1.0;
}
node Apoptosis
{
  logic = OXYGEN & CASP3;
  rate_up = @logic ? 1.0 : 0.0;
  rate_down = @logic ? 0.0 : 1.0;
}
node Survival
{
  logic = OXYGEN & NFkB;
  rate_up = @logic ? 1.0 : 0.0;
  rate_down = @logic ? 0.0 : 1.0;
}
```

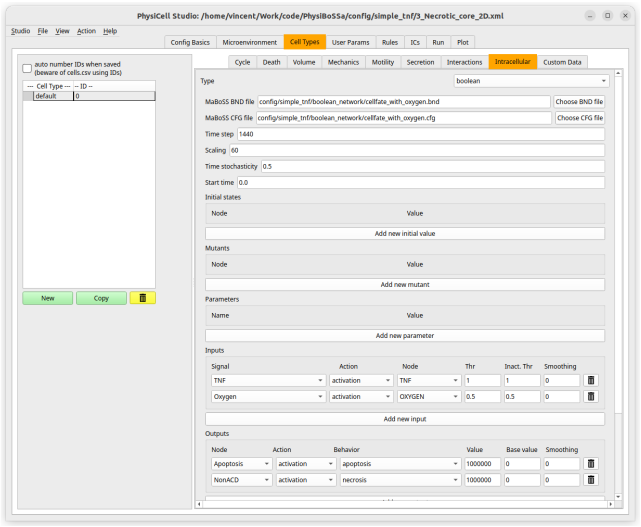


Figure S17: Extract from the MaBoSS BND file, modifications for the addition of the effect of oxygen are highlighted in bold (left). Visualization of the graphical interface for the intracellular definition, with the new mapping for the oxygen (right)

Then we need to modify the MaBoSS model (with any text editor) to add the effect of the Oxygen (or lack thereof) in the intracellular model. In the BND file, we add a new input node for the Oxygen, whose value is fixed during the simulation. We then modify the condition to activate the node *NonACD* (necrotic phenotype), to describe that it can be active not only with a lack of ATP but also with a lack of Oxygen. The formula becomes:
$NonACD = !OXYGEN \mid !ATP$
We also modify the formulas for the other phenotypes, *Apoptosis* and *Survival*, to describe that they can only be active when there is oxygen present, in order to avoid having mixed phenotypes. Their formulas become:
$Apoptosis = OXYGEN \ \& \ CASP3$
$Survival = OXYGEN \ \& \ NFkB$
An extract of this new BND file can be visualized in Figure S17, left panel. Once the BND file is modified, we need to modify the CFG file to set *OXYGEN* initial value to 1 (the model starts with oxygen, but will start lacking it during the simulation). We also add $OXYGEN.set\_internal = 0$ to specify that we

want *OXYGEN* to be an output node, that we can visualize in the simulation results.

With these modifications done, the last thing we need is to create the mapping for the Oxygen, to link the Oxygen from the microenvironment to the *OXYGEN* input node. To do this, we create an input mapping, associating the Oxygen signal to the *OXYGEN* MaBoSS node, with a threshold of 0.5. These new settings can be visualized in Figure S17, right panel. An XML settings file corresponding to this step is available in config/simple_tnf/3_Necrotic_core.xml.

S3.7 Creating IKK++, cFLIP++ mutant

Another specificity of PhysiBoSS is to be able to simulate different clones inside a population of tumor cells. With this example, already presented in the previous publication, we show how two populations with different mutation profiles can be simulated and the impact that a small resistant clone may have following a drug treatment.

To create a mutant population, we first need to create a new cell definition, based on the existing cell definition. For this, we go to the *Cell Types* tab, select our cell definition, and click on the copy button. Then we rename the newly created cell definition to *mutant*. Then, with this mutant cell definition selected, we go to the *Intracellular* sub-tab and add two new mutants by clicking on the *Add new mutant* button twice. We set the first mutant to control the node IKK, and set its value to 1, and the second to control the node cFLIP, and also set its value to 1. These new settings can be visualized in Figure S18.

With this cell definition created, we now need to create a new initial distribution of cells by going to the *ICs* tab. Here, we can choose an arbitrary proportion of the mutant cells in the tumor cell population, as an example we will use one-third of cells harboring the mutations. To do this, we first create a random fill of 535 default cells, on an annulus/disk with (x0, y0) = (0,0), R1 = 0, R2 = 250. Then, we do another random fill, this time with the mutant cell definition and with 265 cells, on the same disk. We save this new initial condition as cells_mutants.csv, in the simple_tnf folder.

Finally, we go back to the *Config Basics* tab, to select our new cells_mutants.csv file in the initial conditions. An XML settings file corresponding to this step is available in config/simple_tnf/4_Mutants.xml.

S3.8 Adding a autocrine secretion of TNF for NF-kB-driven surviving cells

In the case where we want to reproduce the behavior of autocrine secretion of TNF by the resistant cells (with Survival node ON), we need to add a new output mapping, linking the *Survival* node to the behavior `TNF secretion target`.

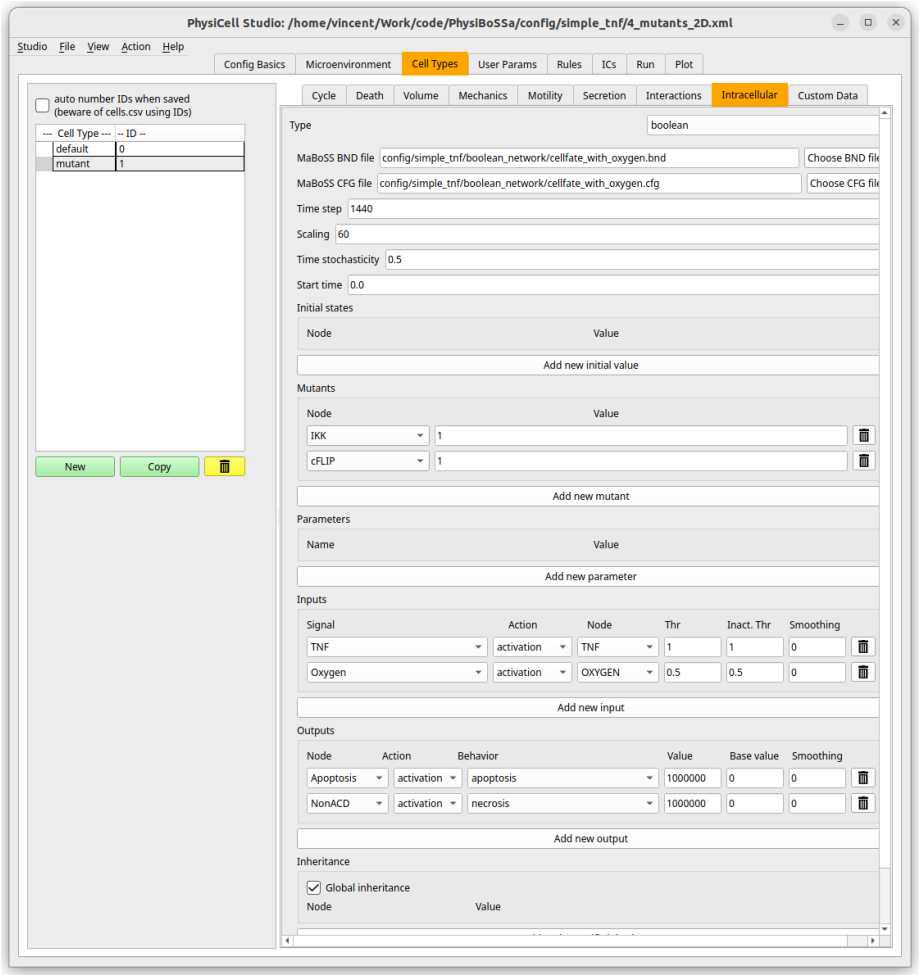


Figure S18: Visualization of the graphical interface for the intracellular definition, showing the definition of the mutant cell lines, with the additional mutations in the intracellular definition.

S3.9 Runtimes

The different versions of this model show an increase in complexity of the model, together with an increase in their simulation times (from 33h to 200h). Their runtimes also increase, from 1 minute for the initial setup to 7 minutes for the version with the necrotic core. The mutant version shows a completely different runtime, as it simulates an exponential growth once the mutant becomes dominant, which goes up to 50 minutes due to the huge amount of cells at the end of the simulation. This shows the dependency of the runtime on the amount of

cells. Simulations were performed on a laptop (Intel i7, up to 4.70 GHz, using 6 cores).

S4 Cell cycle model

This model can be seen as an alternative to the cell cycle in a PhysiCell model with molecular details about the processes. It aims to reproduce the temporal activation of the cell cycle phases, based on the transient oscillations of the intracellular model taken from [3]. To set up the model, no custom C++ code is required. The user can start building it from the template model provided by PhysiCell/PhysiBoSS. The only major requirement is the intracellular .bnd and .cfg files, which contain the adapted MaBoSS model from Sizek and colleagues, available at this link: `https://github.com/PhysiBoSS/PhysiBoSS/tree/master/sample_projects_intracellular/boolean/tutorial/config/cell_cycle/boolean_network`. The analysis of the intracellular model has been provided in the script folder of the tutorial project.

S4.1 Cell cycle choice and parameter setting

For this case study, we are not interested in making changes in the *Config Basics* tab. Instead, we wish to re-create the cell cycle model presented in the main text, we will work mainly in the *Cell Types* tab. Once clicked on the *Cell Types* tab, we can visualize and select the different cell cycle models for our simulation. To choose the most appropriate one, we first have a look at the MaBoSS analysis of the model from Sizek and colleagues.

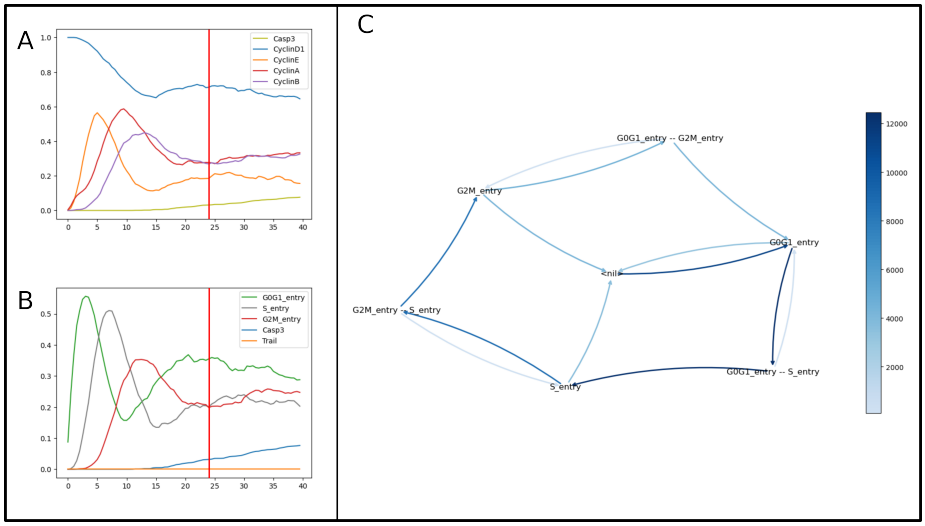


Figure S19: MaBoSS analysis of the Sizek model.

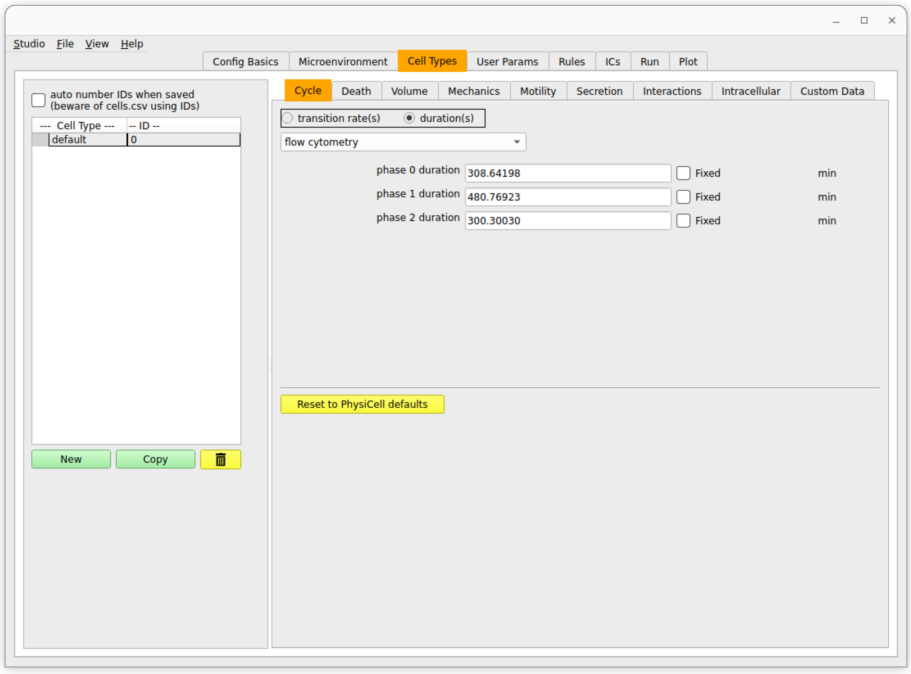


Figure S20: *Cell types* tab. In this tab, the user can add, remove, or modify the cell types.

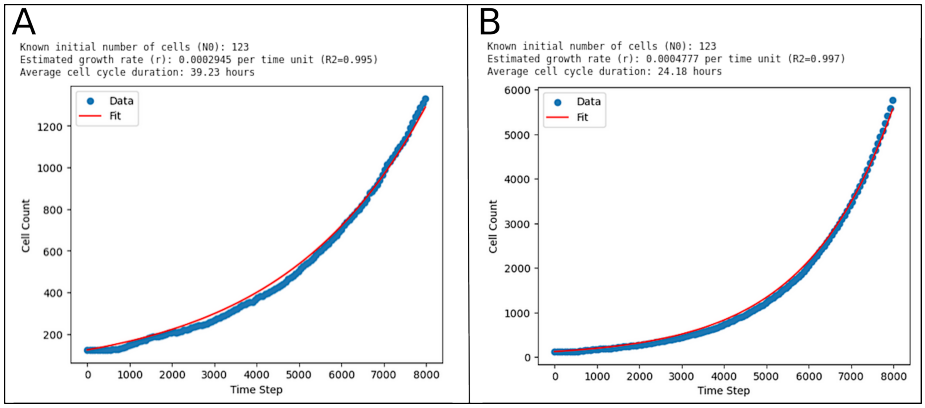


Figure S21: Analysis of the population growth for different values of the scaling parameter. We fit the data to follow an exponential growth curve

From the analysis of the model, as shown in Figure S19 panel A, it is possi-

ble to observe the sequential activation of the Cyclins (CyclinE, CyclinA, CyclinB, and CyclinD1). We decided to use those four nodes to characterize the read-out nodes that will determine the cell cycle transitions. We decided to split the cell cycle into 3 phases and created 3 read-out nodes, G0G1_entry, S_entry, and G2M_entry. We modified rates and rules for these nodes to obtain a clean sequential activation of these phases. This is confirmed by the state transition graph in Figure S19 panel C. From this graph it is possible to observe that a single cell from a <nil> state (where none of the read-out nodes are active) the most probable sequence of state activation is G0G1_entry, G0G1_entry–S_entry, S_entry, G2M_entry–S_entry and finally G2M_entry (See notebook Cell_cycle_boolean_analysis.ipynb).

Now that we have the read-out nodes set, we select the more appropriate cell cycle model from PhysiCell to associate. There are 6 default cell cycle models to choose from, but those divided into three phases are only two: the advanced Ki67 model and the flow cytometry model. We decided to select the second one (Figure S20). The model has a total duration of 18 hours according to the default rates provided by PhysiCell. For our case, we want the intracellular model to define the duration of each phase. We proceed then to set the rate of each phase to 0 (or the duration to a very high number). In doing so, we make sure that a single agent will be stacked in one of the phases until the intracellular model allows the switch to the next cell cycle phase. Similarly, we move to the *Death* sub-tab and set the apoptotic death rate to 0 so that the caspases activity in the intracellular model will determine the apoptotic cell death.

S4.2 Coupling the intracellular and the agent-based model

Once the cell cycle and the apoptosis rates are set, we can move to the *Intracellular* sub-tab to start the coupling between the Boolean model and the agent-based one. In this tab, shown in Figure S22, we are going to select the intracellular MaBoSS model and set the intracellular time step, the scaling, the initial conditions, and finally the outputs. First, using the corresponding buttons, we select the .bnd and .cfg files that we previously downloaded and move to the config folder, as shown in Figure S22A. Next, through the `scaling` and the `intracellular_dt` parameter, we decide how often the Boolean model is updated. As explained in the main text, we want to capture the transient behaviors of the intracellular model. First, it is important to extract from the MaBoSS simulation the duration of a single cell cycle. Using the point of minimum of the G2M phase as a reference, we selected 24 time units. So to translate this MaBoSS duration into a cycle of 24 hours in PhysiBoSS, we initially set the scaling value to 60.

To compute the average cell cycle duration in PhysiBoSS, we model cell growth using the exponential function:

$$N(t) = N_0 \cdot e^{r \cdot t}$$

where $N(t)$ is the number of cells at time t, N_0 is the initial number of cells,

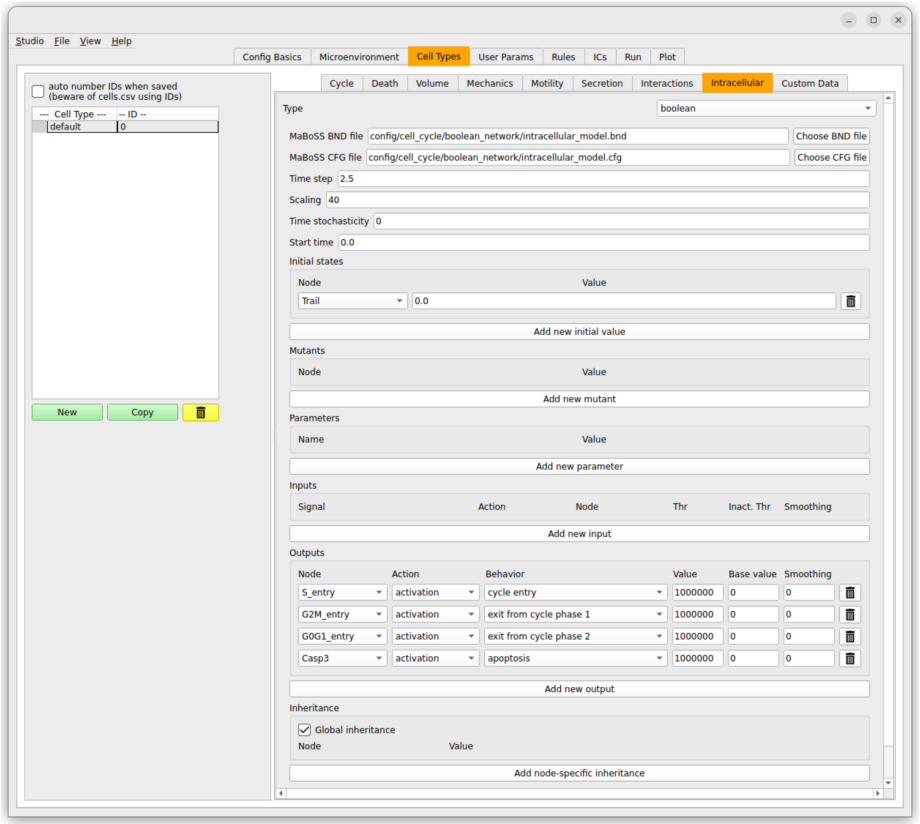


Figure S22: Settings of the *Intracellular* sub-tab to couple the adapted Boolean model from Sizek to PhysiCell through the mapping system

and r is the growth rate. By fitting this model to the simulation data, we obtain the growth rate r. The average cell cycle duration T_{avg} is then calculated as:

$$T_{\text{avg}} = \frac{\ln(2)}{r}$$

This computation is implemented in the Cell_Cycle_Analysis.ipynb notebook. However, and mainly due to the presence of incomplete cycles described in the Boolean analysis of the model (main text), the cell cycle duration we obtain after setting our model as such is 39 hours, as measured in Figure S21. Fortunately, this gives us a value of how much the average cell cycle was slowed down when adding it into PhysiBoSS and allows us to compute a correct value for the scaling parameter. We correct its value to 38, which gives us a cell cycle duration of 24 hours, as seen in Figure S21B.

If no initial conditions are specified, PhysiBoSS will automatically use the

one specified in the .cfg file of MaBoSS. To explore different initial conditions, we can select in the *Initial states* section, the input of the model, like the node Trail (death signal) and GF (growth factor). For a cell proliferative condition, we set the Trail node value to 0 and the GF node value to 1. Finally, since we are not interested in coupling any input with microenvironmental conditions, we link the outputs of PhysiCell (in this case the cell cycle switches) with the nodes of the MaBoSS model. The behaviors we are interested in are:

1. "Cycle entry" or "exit from cycle phase 0", which controls the switch from phase 0 (G0G1 phase) to phase 1 (S phase)
2. "exit from cycle phase 1", which controls the switch from phase 1 (S phase) to phase 2 (G2M)
3. "exit from cycle phase 2", which controls cell division and the switch from phase 2 (G2M) to phase 0 (G0G1)
4. "apoptosis", which triggers the death of the cell

We proceed to connect the node S_entry to the behavior "Cycle entry", G2M_entry to "exit from cycle phase 1", G0G1_entry to "exit from cycle phase 2" and finally the node Casp3 to the behavior "apoptosis".

S4.3 Setting the initial position of the cells

If not specified, PhysiCell by default, will automatically place randomly in the domain, 5 cell types for each cell type. The number of cells placed can be modified in the "User Params" tab, through the parameter *number_of_cells*. Otherwise, it is possible to specify the cell position in the *ICs* tab (Initial Conditions). More information about cell placement can be found in the PhysiCell Studio publication at (`https://github.com/PhysiCell-Tools/PhysiCell-Studio`).

Now that everything is coupled and set, we can move to the *Run* tab and run our simulation.

S4.4 Setting different initial conditions

With this model, we can explore the effect of different mutations (as reported in the main text) as well as explore different initial conditions. Those can be set in the *Intracellular* sub-tab (where the MaBoSS functionality is set). For example, we can link three initial conditions to reproduce Figure 3, graph C in the original publication from Sizek et al. as shown in Figure S23. In the Initial States table, we select three nodes: Trail, GF, and GF_High. We can now set different values for each node, and explore their effects on the simulation (see Figure S23 for details). To add new mutants, you need to select the desired node in the Mutants table. The three examples described in the main text, PLK1++, Foxo3++, and p110- - are available in the `config/cell_cycle` folder.

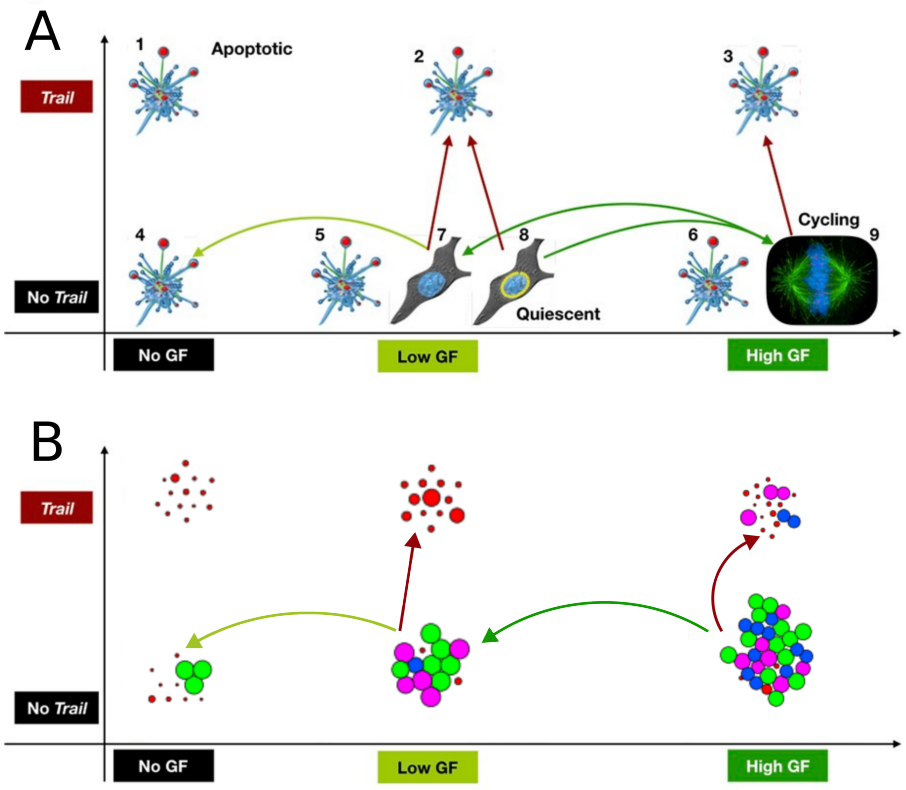


Figure S23: According to the initial conditions, it is possible to fall into different scenarios, reproducing the results from Sizek et al. (Figure adapted from [3])

S4.5 Runtimes

This model simulates a small population, starting at 13 cells and culminating at 73 cells (for the p110 mutant). Thanks to this, the runtime stays very small, around four seconds. Simulations were performed on a laptop (Intel i7, up to 4.70 GHz, using 6 cores).

S5 T-cell differentiation upon Dendritic cell stimuli model

This model simulates the behaviors of three distinct cell populations, lymphatic endothelial cells (LECs), naive T cells, and dendritic cells (DCs). In particular, it shows how the DCs are attracted to the lymph node by the CCL21 secret by the LECs, and how the naive T cells can differentiate into three different

sub-cell types (Th1, Treg, and Th17) upon contact with the DCs. For more information about the biology behind this process, please see the main text. The intracellular Boolean model of the naive T cells is adapted from Corral-Jara et al. [4], while the model for the dendritic cells is a toy model built for this example, as shown in Figure S24. The user can start building it from the template model provided by PhysiBoSS. The only major requirement is the intracellular .bnd and .cfg file, which contains the adapted MaBoSS models. The user can find both models at the following link: `https://github.com/PhysiBoSS/PhysiBoSS/tree/master/sample_projects_intracellular/boolean/tutorial/config/differentiation/boolean_network`. The analysis of the intracellular model has been provided in the script folder of the tutorial project. To set the project, see S2.

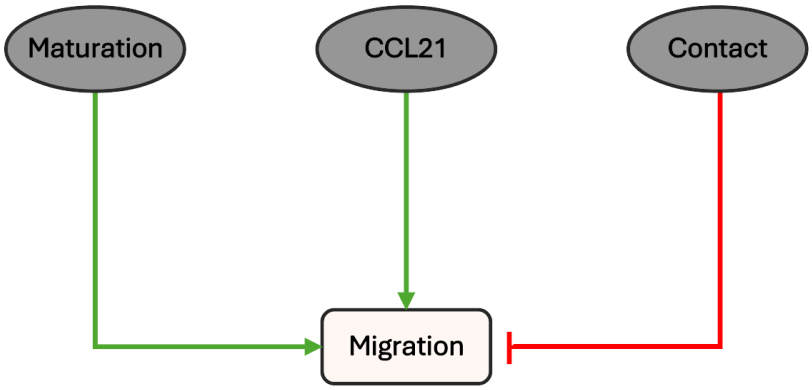


Figure S24: Toy model network for the dendritic cells. While this network does not provide any useful insight into dendritic cell's biology, it can be extended to add significant pathways.

S5.1 Substrate configuration

For this case study, we are not interested in making changes in the *Config Basics* tab. We start setting the project from the *Microenvironment* tab. Here we create the substrate that will act as a chemoattractant from the dendritic cells and will be secreted by the endothelial cells. By default, an example of a substrate is already set. We will then simply rename it and change its parameters. We can double-click on the name of the substrate, on the left, and change its name to "CCL21". We assign a low value of diffusion coefficient, 1000 micron^2/min, and a decay rate of 0.005 1/min, chosen so that the gradient will reach a value of 1.0 mmHG (necessary to trigger the input signal) in the proximity of the dendritic cells, after ≈ 10 time steps (around 6 simulated hours). We set all the other parameters to 0 since our substrate does not require any initial condition.

S5.2 Creating the cell types

This model encompasses a total of 6 different cell types, two of which are characterized by two different MaBoSS intracellular models. To begin with, we are going to create all the cell types and then modify them one by one. By default, PhysiCell provides the user with a default cell type. We renominate it "T0", then we click on the "New" button, creating 4 additional cell types. We renominate them Treg, Th1, Th17, and dendritic_cell. Finally, we create an additional cell type, the endothelial cell, which will be in charge of secreting the chemoattractant for the mDC. We are now ready to personalize each cell type as follows:

S5.2.1 T0 and intracellular coupling

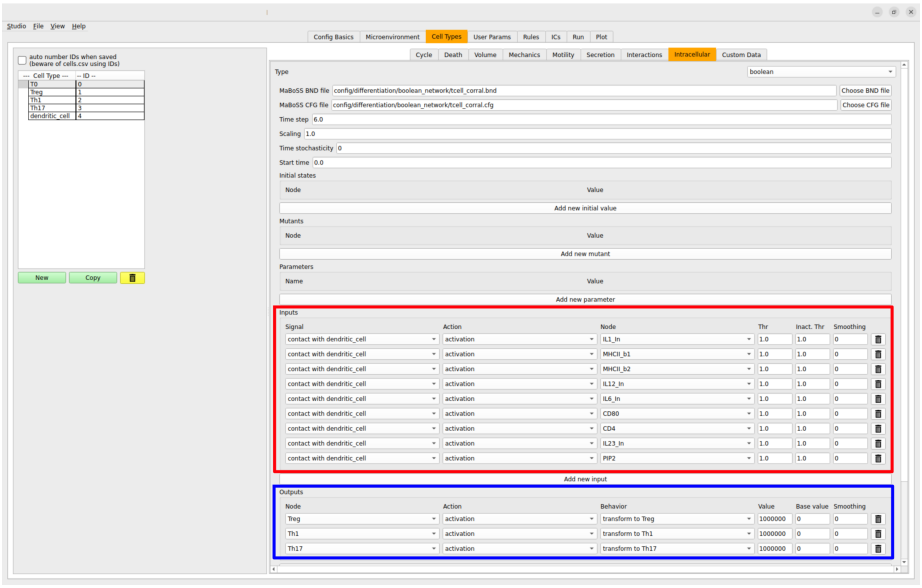


Figure S25: *Intracellular* sub-tab with settings from the cell types T0 of the T cell differentiation model. The red rectangle indicates the inputs to set, and the blue one the outputs.

The naive T cell population has no special behavior except to differentiate when they come in direct contact with dendritic cells in their active state. Their configuration is quite simple:

- In the *Cycle* sub-tab, we select the "live cells" cell cycle model and set the rate to 0.

- In the *Death* sub-tab we set to 0 the rates for Apoptosis and Necrosis.

In this way, the T0 cell type will not proliferate nor die.

- In the *Motility* sub-tab, we simply disable the motility, unchecking the box "enable motility".

With this setting, the T0 will not move around the domain but will stay in place secreting CCL21 and waiting for the signal from the dendritic cells. Next, we couple the Boolean model with the dictionary of signals/behaviors in the *Intracellular* sub-tab:

- We make sure the Boolean intracellular type is selected, allowing us to select the .bnd and .cfg MaBoSS files to couple.
- For the T0 cell, we choose the tcell_corral.bnd and .cfg files that we previously downloaded in the config folder.
- As shown in the MaBoSS analysis, we are interested in the stable states of the model. We set a scaling of 1 and a time step of 6, to give enough time to the MaBoSS simulation to reach a stable state.
- The T0 will react to only one signal: "contact with dendritic_cell". This signal will activate many inputs of the intracellular model, in particular: IL1_1In, MHCII_b1, MHCII_b2, IL12_In, IL6_In, CD80, CD4, IL23_IN and PIP2 (see Figure S25 for details).
- The output of the Boolean model will determine the result of the differentiation. We set as output:
  - Node: Treg, Action: activation, Behavior: transform to Treg.
  - Node: Th1, Action: activation, Behavior: transform to Th1.
  - Node: Th17, Action: activation, Behavior: transform to Th17.

It is important to note that the previous behaviors will not be shown if the relative cell type is not previously created.

S5.2.2 Treg, Th17, Th1

Those cell types have no particular behaviors and necessitate no coupling with any intracellular MaBoSS model. As for the T0 cell types, we just set the Cell Cycle and Death models:

- In the *Cycle* sub-tab, we select the "live cells" cell cycle model.
- In the *Death* sub-tab we set to 0 the rates for Apoptosis and Necrosis.

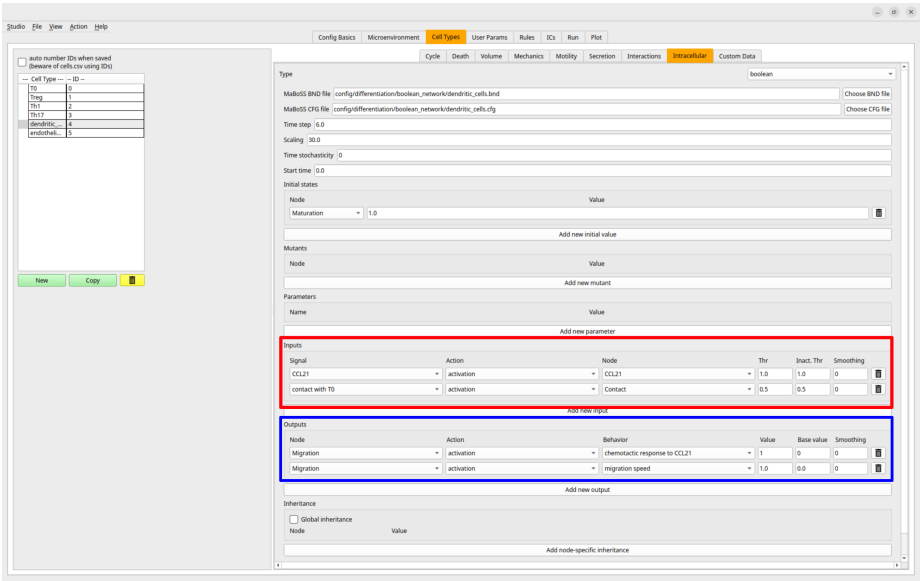


Figure S26: *Intracellular* sub-tab with settings from the cell types dendritic_cell of the Tcell differentiation model. The red rectangle indicates the inputs to set, and the blue one the outputs.

S5.2.3 Dendritic cells and intracellular coupling

The dendritic cells are the only motile cell types in the simulation. Their function is to deliver the antigen signal to the lymph node to start the naive T-cell differentiation. We created for this cell type an intracellular ad-hoc model that can be substituted in the future with a proper Boolean model. Their configuration is the following:

- In the *Cycle* sub-tab, we select the "live cells" cell cycle model and set the rate to 0.
- In the *Death* sub-tab we set to 0 the rates for Apoptosis and Necrosis.
- In the *Motility* sub-tab, we make sure the motility and chemotaxis are enabled. We set the migration bias to 0.8. This value ensures that mDCs reach the naive T cells in a relatively short time while their motility keeps some degree of stochasticity. The speed for the motility will be initialized later by the intracellular model.

Similarly to the other cell types, the dendritic cells will not grow, divide, or die during the simulation. They will be attracted by the CCL21 gradient when the intracellular model sets the right behaviors. We then move to the

coupling of the MaBoSS model with the dictionary of signals/behaviors in the *Intracellular* sub-tab as shown in Figure S26:

- We make sure the Boolean intracellular type is selected, allowing us to select the .bnd and .cfg MaBoSS files to couple.
- For the dendritic cells we choose the dendritic_cells.bnd and .cfg files that we previously downloaded in the config folder.
- As for the T0, we set the scaling to 1 and intracellular time to 0, since we are interested in the stable states of the model.
- The intracellular model is a simple phenomenological representation of the genetic regulation inside the dendritic cells. Nevertheless, we can acknowledge the maturation state of this cell type by setting the initial condition of the value for the node Maturation to 1.
- The dendritic cells will react to two different signals:
  - Signal: CCL21, Action: activation, Node: CCL21, and Thr (threshold) 1.0 .
  - Signal: contact with T0, Action: activation, Node: Contact, and Thr 0.8.
- The output of the Boolean model will determine the motility and target of the dendritic cell:
  - Node: Migration, Action: activation, Behavior: chemotactic response to CCL21.
  - Node: Migration, Action: activation, Behavior: migration speed with value 1.

S5.2.4 Endothelial cell

The endothelial cell has no particular behaviors, except secreting CCL21 which will spread in the domain attracting the mDC cells toward the location of the naive T-cell. The configuration is similar to the T0 cell type but without any intracellular model.

- In the *Cycle* sub-tab, we select the "live cells" cell cycle model and set the rate to 0.
- In the *Death* sub-tab, we set to 0 the rates for Apoptosis and Necrosis.

In this way, the endothelial cell will not proliferate nor die.

- In the *Motility* sub-tab, we simply disable the motility, unchecking the box "enable motility".

Finally, we move to the *Secretion* sub-tab, select the substrate CCL21, and set the secretion rate and the target to 10.

S5.3 Setting the initial position of the cells

If not specified, PhysiCell by default, will automatically place randomly in the domain, 5 cells for each cell type. The number of cells placed can be modified in the *User Params* tab, through the parameter *number_of_cells*. Otherwise, it is possible to specify the cell position in the *ICs* tab (Initial Conditions). For this model, we decided to create an initial population of 7 dendritic cells in the bottom right corner of the domain, and a population of 85 T0 in the top right corner of the domain. Finally, we placed a single endothelial cell in the top left corner of the domain (-300, 290).

Now that everything is coupled and set, we can move to the *Run* tab and run our simulation.

S5.4 Modulating the differentiation through the intracellular model

It is possible to intervene in the model by modifying the physical parameters of the cell (like the migration bias, as reported in the main text), or introducing mutations. For this purpose, we tested the model of Corral-Jara and colleagues, to look for possible mutations that do not directly trigger the differentiation of the T0, but make them more vulnerable to the input delivered by the mDCs.

We conducted at first a sensitivity analysis to test each possible single mutant and selected the ones that do not trigger any differentiation of the T0. We proceed then to test the input delivered by the mDCs directly on the aforementioned states. We individuated in this way a series of single mutations that modulate the differentiation of the T0 cells.
Figure S27 shows two examples of this: panel A shows an inhibiting mutation of NFKB, leading to a full Treg differentiation; panel C shows an inhibiting mutation of FOXP3_2 that leads to a total absence of Treg differentiation, accentuating the differentiation into TH1 and TH17. Both those results can be obtained by adding a mutant in the *Intracellular* sub-tab, and setting the corresponding node value (in this case NFKB or FOXP3_2) to 0.
Using the MaBoSS parameters, it is possible to modulate the effect of those mutations by modifying the transition rates for each node. Similar to what was done for the mutation simulations, we can select a new parameter by clicking on the button *Add new parameter*. Among the possible parameters to select, some of them are identified by the prefix "$u_" or "$d_" followed by the name of one of the nodes of the intracellular model. Those parameters control respectively the activation and inhibition rate of the node. Setting the activation rate of a node to 0 is equivalent to performing an inhibiting mutation; in the same way, setting the inhibition rate of a node to 0 is equivalent to performing an activating mutation. As shown in Figure S27 panel B and D, by assigning different values to "$u_NFKB" and "$u_FOXP3_2", it is possible to control more finely the differentiation into Treg.

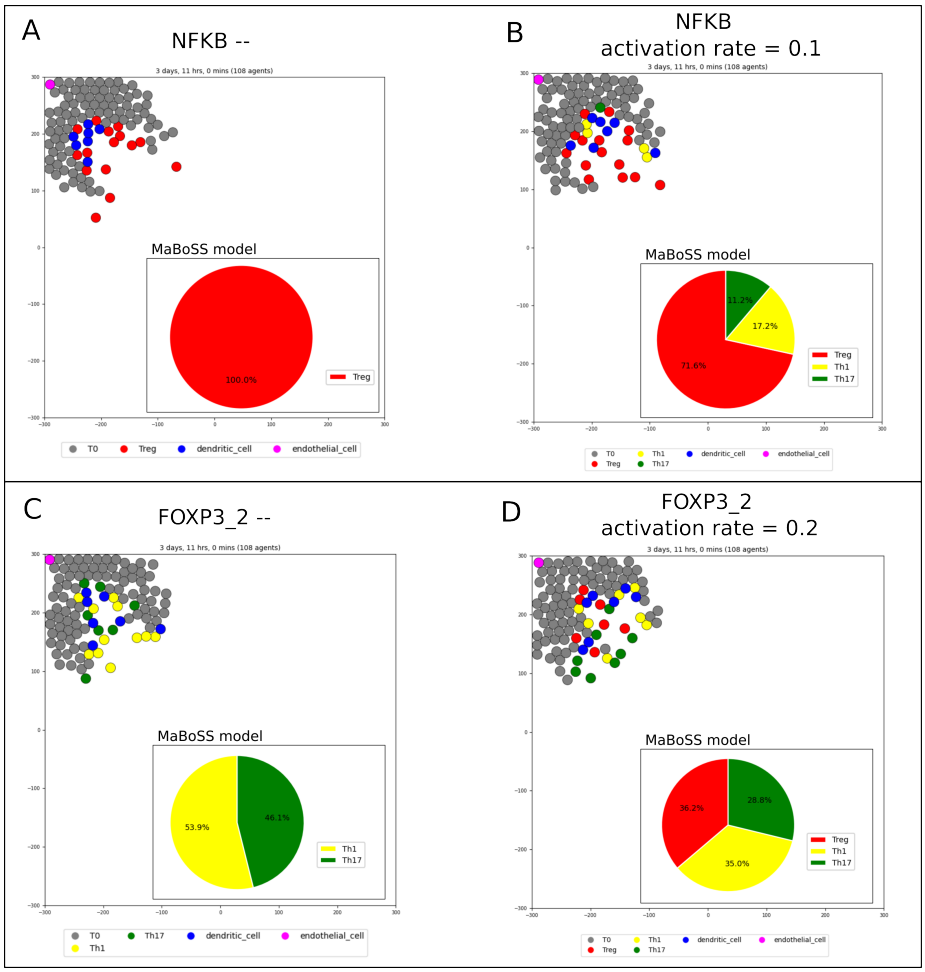


Figure S27: Two different mutations affect the differentiation dynamic. Each panel showcases a PhysiBoSS simulation and the relative simulation of the MaBoSS model for the T0 cells. Panel A shows an inhibiting mutation of NFKB leading to a full Treg differentiation. In panel B, the differentiation into Treg is mitigated by setting a low value of the activation rate of the node NFKB, instead of completely inhibiting it. Panel C shows an inhibiting mutation of FOXP3_2 (level 2) leading to a differentiation state where the Treg cell type is absent. Panel D shows how setting the activation rate of FOXP3_2 to a low value, slowly reintroduces the Treg cell type.

S5.5 Runtimes

This models stays very simple, using only one substrate and around a hundred cells. Thanks to this setup, the simulation time stays very small, a little less than a minute on a laptop (Intel i7, up to 4.70 GHz, using 10 cores).

S6 Cell invasion model through extracellular matrix

This model follows the one previously published in [5]. In this scenario, an initial population of epithelial cells is surrounded by the extracellular matrix (ECM), a non-diffusive substrate that acts as a barrier, preventing the cells from moving towards the source of oxygen. The cells, to invade, need to acquire the proper mesenchymal phenotype, secrete matrix-metalloproteases (MMPs), and adhesion to the ECM. The differences between this new version of the model and the previous one are displayed in the main text. The model cannot be built from the template_BM project, since it requires some custom code to add repulsion to the ECM substrate. The custom code can be found at: `https://github.com/PhysiBoSS/PhysiBoSS/blob/master/sample_projects_intracellular/boolean/cancer_invasion/custom_modules/custom.cpp` (to note, both the custom.cpp and custom.h are required). The custom files must be placed in the custom_modules folder (as shown in S2) before compiling the project.

S6.1 Update and development of the model

Since it was developed for an earlier version of PhysiBoSS, the intracellular model comes ready to plug in. However, many improvements and new features have been added to the latest PhysiCell/PhysiBoSS version, including an improved algorithm for simulating elastic adhesion between cells. For this reason, the code used to simulate the different physical interactions between cells and substrates has been reduced to make the most of the new functions available in PhysiBoSS. However, the part of the code that allows agents to adhere to and be repelled by the ECM is retained. There, the amount of cell-to-cell and cell-to-ECM contact as well as nucleus deformation, is calculated. For this new updated version of the model, we decided to split epithelial and mesenchymal cells in two cell types. Each cell type has its cell definition with its parameters but shares the same intracellular model. Both cell types use the advanced Ki67 cell cycle model, provided by PhysiCell. Epithelial and mesenchymal cell type differs in motility and adhesion properties: in general, epithelial cells are characterized by more rigid adhesive properties and are not motile, unless pushed or trailed by other mesenchymal cells. The latter instead, have less adhesive more motile properties, and according to the phenotype dictated by the intracellular network, can move following a random walk or the oxygen gradient. Playing around those two properties, adhesion and motility, it is possible to shift the

model towards a representation of migrating single cells (Figure S28A) or cluster formation (Figure S28B).

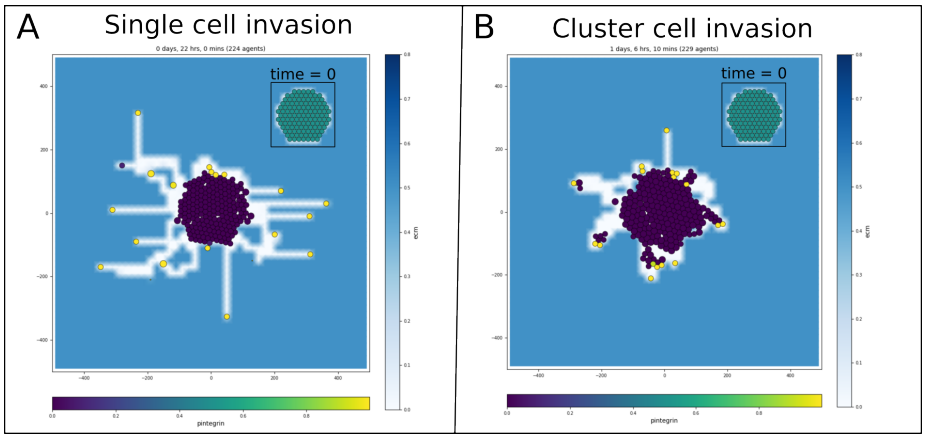


Figure S28: Simulation of an epithelial cell mono-layer surrounded by extracellular matrix (ECM) starting from the same initial conditions and characterized by different bio-physical parameters. The color bar at the bottom indicates the percentage of integrins recruited for each cell, the one at the right represents the amount of ECM in each voxel of the microenvironment. Mesenchymal cells in panel A are characterized by a lower adhesion elastic constant, leading to single-cell migration. Mesenchymal cells in panel B are characterized by a higher adhesion elastic constant, causing the formation of clusters of migrating cells.

In the following sections, we will cover how to set up the project using PhysiCell Studio, taking into account that the custom code is already integrated.

S6.2 Setting the substrates of the microenvironment

As already mentioned in the main text, the model encompasses 3 different substrates: Oxygen, ECM, and TGF-β. The first one can freely diffuse in the microenvironment and is used as a signal for chemotaxis. ECM and TGF-β are non-diffusive substrates, with no decay rate. The first one acts as a barrier and provides both repulsion and adhesion to approaching cells. The second one is non-diffusive too, but does not provide any physical interaction and is used as a signal for the intracellular pathways for the activation of some mesenchymal phenotypes. To set the substrate, in the GUI, we move to the *Microenvironment* tab and create 3 Substrate that we name oxygen, ECM, and TGF-beta as shown in Figure S29. For each substrate the settings are the following:

1. oxygen: `diffusion coefficient` = 60000.0 micron^2/min, `decay rate` = 0.0005 1/min, `initial condition` = 38.0 mmHg, `Dirichlet BC` = 38.0 mmHG.

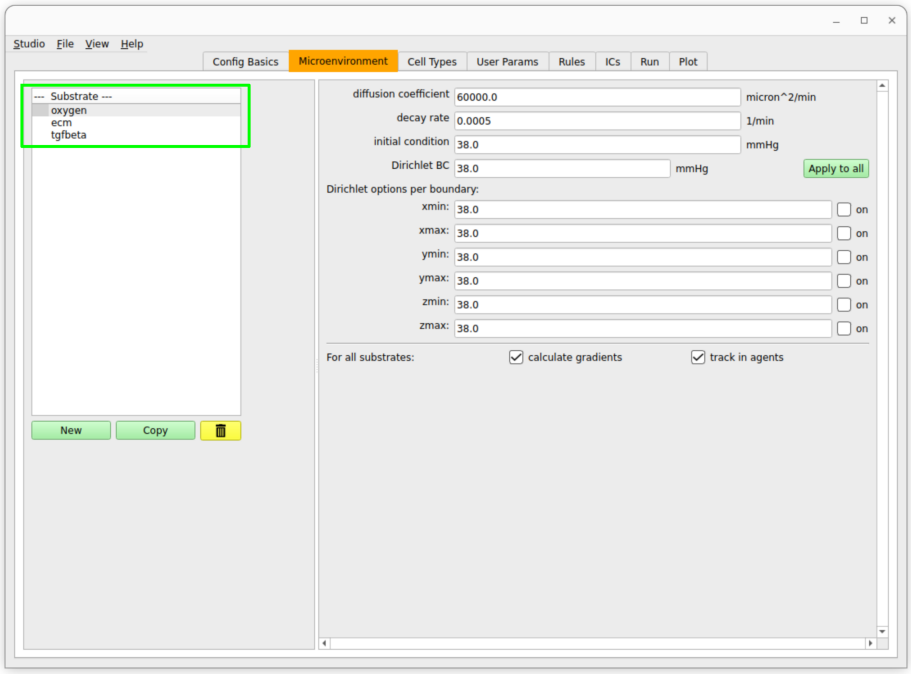


Figure S29: The settings for the substrate in the invasion project

2. ecm: `diffusion coefficient` = 0.0 micron^2/min, `decay rate` = 0.0 1/min, `initial condition` = 0.0 mmHg, `Dirichlet BC` = 0.0 mmHG.

3. tgfbeta: `diffusion coefficient` = 0.0 micron^2/min, `decay rate` = 0.0 1/min, `initial condition` = 0.0 mmHg, `Dirichlet BC` = 0.0 mmHG.

With this setting, we ensure the oxygen will start with a homogeneous initial distribution at a value of 38.0 mmHG and will subsequently diffuse from the borders of the domain towards the center, where the tumor is located. ECM and TGF-β start with the same initial homogeneous condition but there will be no source of diffusion nor any decay.

S6.3 Creating epithelial and mesenchymal cell types

Epithelial and mesenchymal cell types have very different characteristics, both in mechanistic and genetic terms. Epithelial cells have high adhesive properties and are less motile, while mesenchymal cells are less adhesive but acquire high motile properties. Despite sharing the same intracellular model, in the next paragraph, we will illustrate how to characterize each cell type and which signals/behaviors to connect with the inputs/outputs of the Boolean network they both share.

We start by creating, in the *Cell Types* tab, the two cell types, epithelial and mesenchymal.

S6.3.1 Cell cycle and death

The cell cycle selected for both cell types is the advanced Ki67 model, which is composed of three phases and lasts for 20 hours. As per the previous version of this invasion model, we set for both the cell types, the phase 1 rates to 0, blocking it. In fact, the phase 1 rate will be dictated by the intracellular node "Cell_growth" state. Similarly, in the *Death* sub-tab, we set the Apoptosis and Necrosis death rate to 0.

S6.3.2 Mechanics, Motility, and Secretion

As previously stated, epithelial and mesenchymal cell types are characterized by different mechanical properties. According to the combination of parameters used for each cell type, we can reproduce different scenarios, raising the possibility of observing single-cell invasion or cluster formation. This new version of the invasion model comes with two configuration files, available at `https://github.com/PhysiBoSS/PhysiBoSS/tree/master/sample_projects_intracellular/boolean/cancer_invasion/config`, with two separate combinations of parameters, act at reproducing a single cell invasion scenario and a collective migration one. In the *Mechanics* sub-tab, we can set the parameters that influence the hook-like adhesion dynamic between cells. First, to avoid the collapse of the spheroid in one single point, and to avoid too much overlap between cells, we set the value of "cell-cell repulsion strength" to 100. The other parameters that influence the cluster formation are the "elastic constant", "attachment rate" and "detachment rate". We can leave the rest of the parameters at their default value. The "elastic constant" modifies the force of attraction between two cells that have established a hook-like adhesion. If its value is too high, it will increase the rigidity of the initial spheroid of cells, making more difficult the initial cell detachment. A too-low value, on the contrary, will disperse the cells, causing them to migrate more easily without any attachment. The rates for attachment and detachment, influence the possibility to establish or break hook-like adhesion between cells. To increase the chance of cluster formation, we can set the "detachment rate" to zero. In addition, to take into account that mesenchymal cells are characterized by fewer adhesion sites than epithelial ones, we can set different values for "attachment rate" (epithelial: 1.0, mesenchymal: 0,4 for example.). There is no fixed value for these parameters; they exhibit variability across situations.

In the *Motility* sub-tab, we set the values for the cell movement. There is a clear difference between epithelial and mesenchymal cell types: the first one is not able to move spontaneously (can still be dragged or pushed), while the latter can randomly move at low speed.

In the *Secretion* sub-tab, we can set the values of secretion and uptake for oxygen, Tgfβ, and ECM. Both the cell types cannot secrete oxygen, but they

uptake it at a rate of 0.005 unit/min. The same goes for TGFβ, with an uptake rate of 0.02 unit/min. The uptake rate for ECM instead, will depend directly on the state of the related intracellular node.

S6.3.3 Intracellular coupling

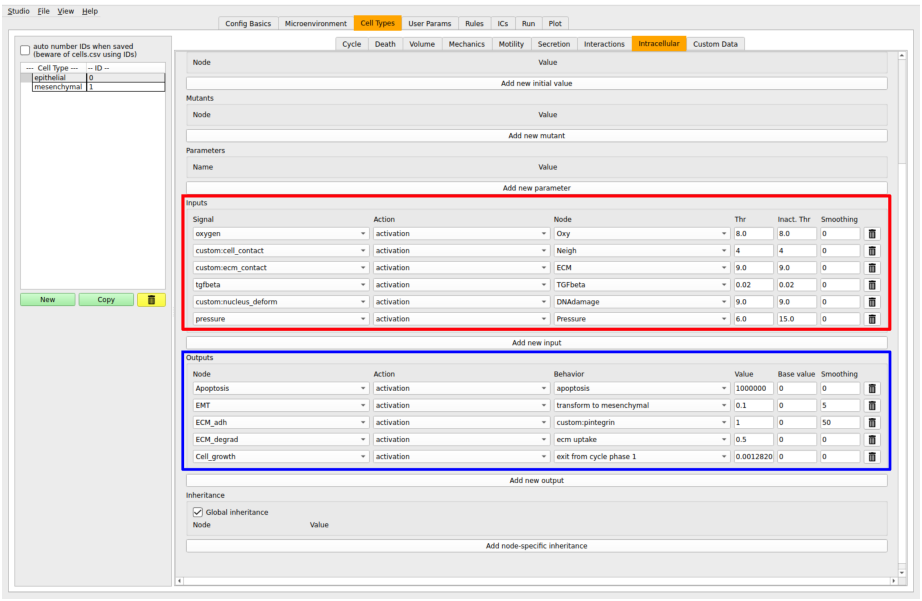


Figure S30: Signals and behaviors coupling for epithelial cell types

Epithelial and mesenchymal cells have different behaviors and react differently to external stimuli. Despite sharing the same intracellular model, the output of the intracellular model is connected differently for the two cell types. As done for the models illustrated in the previous sections, we proceed to connect the MaBoSS intracellular model to the PhysiCell framework, first, setting the intracellular step and the scaling, then, taking advantage of the dictionary of signals/behavior to set input and output nodes.
As previously done with the cell cycle, we are interested in the transient behavior of the MaBoSS model (see supplementary materials for the analysis of the intracellular model). In this case, we set the scaling to 10 and the intracellular time step to 6.

The coupling of the signals is the same for both the cell types as shown in Figure S30 and S31. We connect the signal "oxygen" and "tgfbeta", which collects respectively, the level of oxygen and TGF-beta nearby the cell, to the node *Oxy* and *TGFbeta* node. The "pressure" signal is connected to the *Pressure* node,

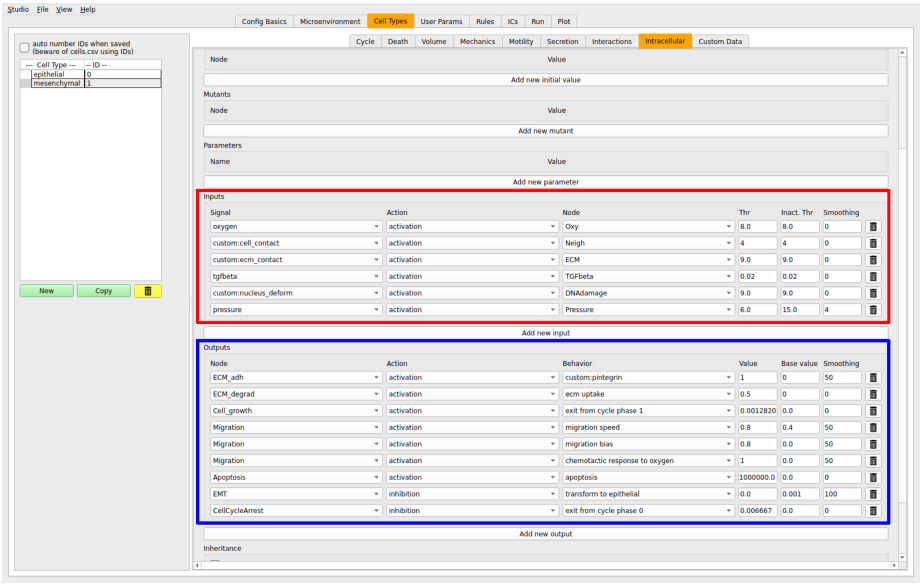


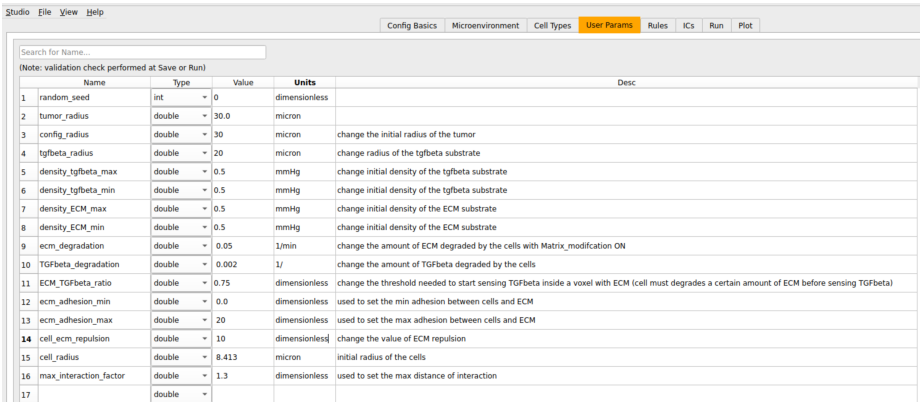
Figure S31: Signals and behaviors coupling for mesenchymal cell types

which ensure that the cell growth is stopped when reaching the critical pressure value. For the "Pressure" signal, the activating and inhibiting thresholds for the related input node are different (Thr: 6.0, Inact. Thr: 15.0). This intentional variability mimics a hysteresis dynamic, illustrating that the cell takes an extended duration to revert to a normal state after being subjected to high pressure. In the custom.cpp code we introduced 3 custom variables, that collect the amount of cell-to-cell contact, cell-to-ECM adhesion, and nucleus deformation. Those three variables, defined inside the custom code of the model, are characterized by the keyword "custom:" (custom: cell_contact, custom: ecm_contact, custom: nucleus_deform). We proceed to connect the three variables respectively to the nodes *Neigh*, *ECM* and *DNAdamage*.

The coupling of the behaviors is quite different in the two cell types. The first thing to notice is that epithelial cells have fewer behaviors coupled than mesenchymal cells. This choice was made to highlight the phenotypic differences between the two cell types. Epithelial cells can die, grow, move (random walk) when not frozen, adhere to the ECM, and differentiate to the mesenchymal cell type. Similarly, mesenchymal cells can die, grow, move (chemotaxis towards oxygen), adhere to the ECM, and differentiate to the epithelial cell type. In addition, they can also increase their speed and bias towards the oxygen gradient as well as degrade the ECM.
All those behaviors are automatically included in the behavior dictionary and can easily linked to the phenotype nodes of the network's model. The node

Apoptosis activates the "apoptosis" behaviors, increasing its transition rate. *Cell_growth* activates the "entry cell cycle" behaviors, allowing the cell to increase its volume and grow. *Migration* increases the "migration speed" and "migration bias" for mesenchymal cells. *Cell_freeze* sets "migration speed" to 0. *ECM_adh* gradually increases the value of the custom variable "pintegrin" from 0 to 1 in a span of 50 time_stap. *ECM_degrad* increases the "ecm uptake" rate. Finally, for the epithelial cell type, *EMT* activates the differentiation to mesenchymal cell type through "transform to mesenchymal" behaviors. For the mesenchymal cell type, *EMT* inhibits the differentiation back to the epithelial cell type ("transform to epithelial").

S6.4 The User parameters



	Name	Type	Value	Units	Desc
1	random_seed	int	0	dimensionless	
2	tumor_radius	double	30.0	micron	
3	config_radius	double	30	micron	change the initial radius of the tumor
4	tgfbeta_radius	double	20	micron	change radius of the tgfbeta substrate
5	density_tgfbeta_max	double	0.5	mmHg	change initial density of the tgfbeta substrate
6	density_tgfbeta_min	double	0.5	mmHg	change initial density of the tgfbeta substrate
7	density_ECM_max	double	0.5	mmHg	change initial density of the ECM substrate
8	density_ECM_min	double	0.5	mmHg	change initial density of the ECM substrate
9	ecm_degradation	double	0.05	1/min	change the amount of ECM degraded by the cells with Matrix_modifcation ON
10	TGFbeta_degradation	double	0.002	1/	change the amount of TGFbeta degraded by the cells
11	ECM_TGFbeta_ratio	double	0.75	dimensionless	change the threshold needed to start sensing TGFbeta inside a voxel with ECM (cell must degrades a certain amount of ECM before sensing TGFbeta)
12	ecm_adhesion_min	double	0.0	dimensionless	used to set the min adhesion between cells and ECM
13	ecm_adhesion_max	double	20	dimensionless	used to set the max adhesion between cells and ECM
14	cell_ecm_repulsion	double	10	dimensionless	change the value of ECM repulsion
15	cell_radius	double	8.413	micron	initial radius of the cells
16	max_interaction_factor	double	1.3	dimensionless	used to set the max distance of interaction
17		double			

Figure S32: List and description of all the custom parameters introduced by the user

Through the graphic interface of PhysiCell Studio, it is possible to manipulate and create different scenarios, as well as implement and high number of behaviors for the cells. Nevertheless, the user can introduce new and advanced functionality into PhysiCell/PhysiBoSS. Such functionalities can be placed in the custom.cpp file. All the parameters linked to those functionalities can be specified in the *User Params* tab, and later in the code, they can be retrieved with the following syntax:

```
parameters.doubles("custom_data");
```

The invasion project showcased in [5] and this new version, introduce a functionality that is not present in the original PhysiCell, that is the physical interaction between a substrate (in this case, the ECM) and a cell. The majority of the parameters present in figure S32, are linked to this functionality and can modify the dynamics between cell and ECM, like the repulsion force or the adhesion. Moreover, for this model, we added a function that randomizes

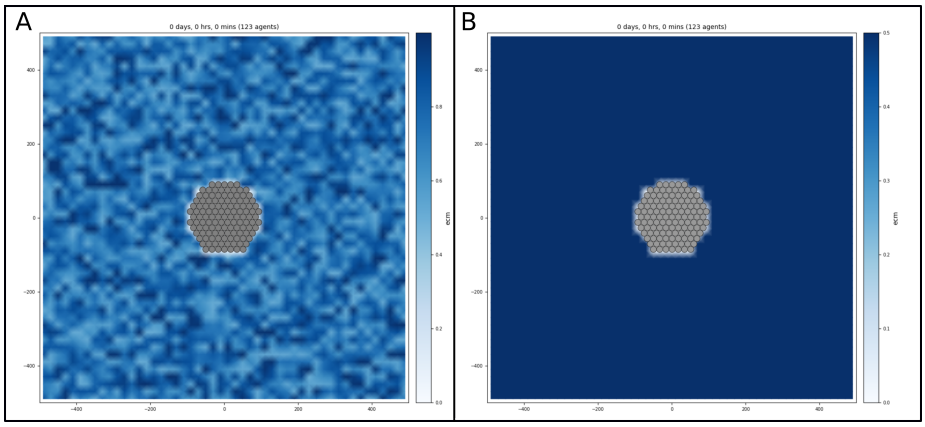


Figure S33: Example of different value for the custom parameter "density_ECM_max" and "density_ECM_min".

the amount of substrate in each voxel of the domain, as shown in figure S33. More details about those functionalities can be found in the custom.cpp file at `https://github.com/PhysiBoSS/PhysiBoSS/tree/master/sample_projects_intracellular/boolean/cancer_invasion/custom_modules`.

S6.5 Setting the initial position of the cells

For this model, we decided to create an initial population of 123 epithelial cells, which is obtained by hexagonal placement of the agents in a disk of radius 100 with center at x, y = 0. Notice that independently of where the agents are placed, a custom function of the model will automatically remove ECM and TGF-β from the voxel occupied by the cells.

Now that everything is coupled and set, we can move to the *Run* tab and run our simulation.

S6.6 Visualizing the output

As per the previous models, once the simulation finished running, we can visualize different kinds of outputs. As shown in figure S34 panel A, we can monitor the degradation of the ECM over time, as well as the boolean state of a node in the cells. Similarly, we can visualize the oxygen gradient and the value of many parameters in the .mat mode (S34 panel B). In the population plot (Panel C figure S34) we can monitor the evolution of the initial epithelial cell population and its differentiation in the mesenchymal type. Finally, we can use the PhysiBoSS state plot to check the Boolean state evolution for each cell of a selected cell type (figure S34, panel D).

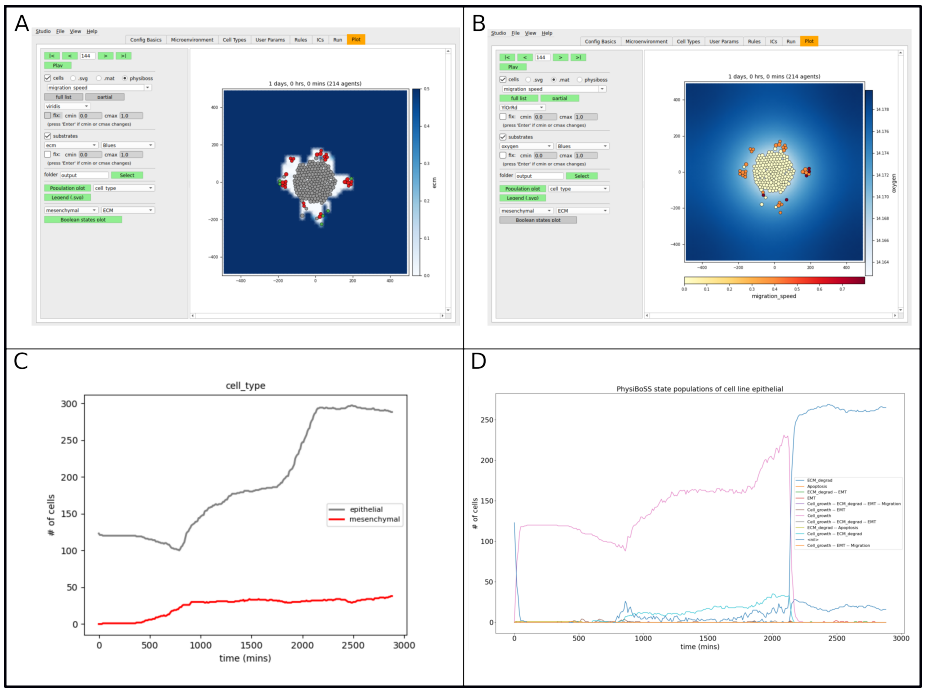


Figure S34: *Plot* tab example for the invasion model. In Figure A the ECM is visualized along with the ECM node activity. Figure B displays the oxygen distribution and the value of migration speed for each cell. Finally, it is possible to plot the evolution of the cell population and the evolution of the Boolean state for each cell in the simulation.

Comparisons and differences with previous version

The new implementation of the model, re-mapped some of the custom parameters and variables, introduced in the previous version, taking advantage of the new features introduced in the latest PhysiCell/PhysiBoSS releases. This drastically diminished the amount of C++ code in the custom.cpp file of the model. In the previous version, the number of parameters defined by the user was around 40, while in the new version is around 14. As shown, the model is fully compatible with Physicell Studio, making all those parameters concerning the ECM plot no longer necessary.
A new feature included in this PhysiBoSS version is a better-defined hook-like adhesion. This kind of adhesion is characterized by an elastic constant which can be different across cell types. This parameter makes the adhesion more rigid or softer. Moreover, a cell type can have different adhesion affinities, making it easier to develop hook-like adhesion with one cell type compared to another one.

The ECM-related parameters remained the same as for the previous version. The model reproduces single-cell and cluster migration through the ECM. Using different combinations of adhesion affinities, elastic constant, and motility properties, it is possible to switch between invasion modes (Figure S28).
The MaBoSS intracellular model allows to intervene in the cell phenotypes, exploring the effect of mutations or reproducing different experimental conditions.

S6.7 Runtimes

All the different versions of this model take under a minute to simulate, using a laptop (Intel i7, up to 4.70 GHz, using 10 cores). More generally, Stack et al. [13] performed code profiling on 3D PhysiCell models (without PhysiBoSS extensions), finding that biological diffusion computations can require over 65% of wall time. For comparison, non-published data showed that PhysiBoSS represents around 1-2% of computation. The original PhysiCell method paper found that execution time scales linearly with the size of the simulation domain, the number of diffusing substrates, and the number of cell agents [8].

References

[1] Calzone L, Tournier L, Fourquet S, Thieffry D, Zhivotovsky B, Barillot E, et al. Mathematical Modelling of Cell-Fate Decision in Response to Death Receptor Engagement. PLOS Computational Biology. 2010 Mar;6(3):e1000702. Publisher: Public Library of Science. Available from: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000702.

[2] Letort G, Montagud A, Stoll G, Heiland R, Barillot E, Macklin P, et al. PhysiBoSS: a multi-scale agent-based modelling framework integrating physical dimension and cell signalling. Bioinformatics. 2019 Apr;35(7):1188–1196. Available from: https://doi.org/10.1093/bioinformatics/bty766.

[3] Sizek H, Hamel A, Deritei D, Campbell S, Regan ER. Boolean model of growth signaling, cell cycle and apoptosis predicts the molecular mechanism of aberrant cell cycle progression driven by hyperactive PI3K. PLOS Computational Biology. 2019 Mar;15(3):e1006402. Publisher: Public Library of Science. Available from: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006402.

[4] Corral-Jara KF, Chauvin C, Abou-Jaoudé W, Grandclaudon M, Naldi A, Soumelis V, et al. Interplay between SMAD2 and STAT5A is a critical determinant of IL-17A/IL-17F differential expression. Molecular Biomedicine. 2021 Apr;2(1):9. Available from: https://doi.org/10.1186/s43556-021-00034-3.

[5] Ruscone M, Montagud A, Chavrier P, Destaing O, Bonnet I, Zinovyev A, et al. Multiscale model of the different modes of cancer cell invasion. Bioinformatics. 2023;39(6):btad374.

[6] Heiland R, Bergman DR, Lyons B, Cass J, Rocha HL, Ruscone M, et al. PhysiCell Studio: a graphical tool to make agent-based modeling more accessible. Gigabyte. 2024. Available from: https://gigabytejournal.com/articles/128.

[7] Metzcar J, Wang Y, Heiland R, Macklin P. A review of cell-based computational modeling in cancer biology. JCO clinical cancer informatics. 2019;2:1–13.

[8] Ghaffarizadeh A, Heiland R, Friedman SH, Mumenthaler SM, Macklin P. PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems. PLOS Computational Biology. 2018 Feb;14(2):e1005991. Publisher: Public Library of Science. Available from: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005991.

[9] Johnson JA, Stein-O'Brien GL, Booth M, Heiland R, Kurtoglu F, Bergman D, et al. Digitize your Biology! Modeling multicellular systems through interpretable cell behavior. bioRxiv. 2023:2023–09.

[10] Stoll G, Viara E, Barillot E, Calzone L. Continuous time boolean modeling for biological signaling: application of Gillespie algorithm. BMC Systems Biology. 2012 Aug;6(1):116. Available from: https://doi.org/10.1186/1752-0509-6-116.

[11] Stoll G, Caron B, Viara E, Dugourd A, Zinovyev A, Naldi A, et al. MaBoSS 2.0: an environment for stochastic Boolean modeling. Bioinformatics. 2017 Jul;33(14):2226–2228. Available from: https://doi.org/10.1093/bioinformatics/btx123.

[12] Bergman D, Marazzi L, Chowkwale M, Bidanta S, Mapder T, Li J, et al. PhysiPKPD: A pharmacokinetics and pharmacodynamics module for PhysiCell. Gigabyte. 2022;2022.

[13] Stack M, Macklin P, Searles R, Chandrasekaran S. OpenACC Acceleration of an Agent-Based Biological Simulation Framework. Computing in Science & Engineering. 2022;24(5):53–63.