Agent-Based Model of the Immune System

Ian Hay, Daniel Margolis April 25th, 2023

Background

With the advance of artificial intelligence in biotechnology and in particular drug discovery, a major focus is on the individual interactions between a therapeutic and its target receptor or cell. However, human biology is governed by system-level interaction. It has proven immensely difficult to predict the impact of a drug targeting one cell or protein on the behavior of the system. While algorithms have successfully generalized the prediction of protein folding, for example, there is a clear lack of a state-of-the-art model for the complex system of immune cells that govern many disease-relevant pathways and targets [1], [2].

The state of incomplete models of the human immune system is indeed due to its complexity. The human body itself has an estimated 30 trillion cells, and human cells are to date the most complicated unit of life studied. Biologists have modeled interactions between proteins and cells successfully as ordered differential equations (ODEs); these ODEs have proven unscalable to the size of relevant biological systems including the immune system [3]–[6].

Recently, researchers have turned to modeling cells as agents and using decision processes to determine their actions for any given state. These agent-based models are less computationally complex than ODEs and, if implemented correctly, can retain similar levels of functionality while increasing scalability [7]–[10].

The goal of this project is to model immune cells as agents in a world with healthy and infected cells. Their mission is to eradicate infected cells while allowing the healthy cells to thrive. As such, the immune cell population should be as low as possible while still maintaining a presence to attack and kill immune cells. Their decision making should also minimize the potential for off-target cytotoxicity, where they mistakenly target healthy cells as infected. In essence, the decision making of immune cell agents on a local scale should, ideally, lead to healthy cell prosperity on a global scale, modeling what our own immune system does on a daily basis to keep us happy and healthy.

We sought to model several key components of the immune system. The world begins as a state of healthy cells with a probability they reproduce on a given turn and a probability they die. The reproduction probability is marginally greater than the death probability to allow the population to grow from an initial sparse seeding. A proportion of the healthy cells begin as infected. The infected are capable of spreading throughout the healthy cells as well as reproducing and dying at the same initial rate; the healthy cells are unable to stop this spread alone. To aid in their longevity, we introduce effector and helper immune cells. Effector cells are capable of moving and killing infected cells; their attack has an area-of-effect to kill all the immediately adjacent neighbors, if successful. The helper cells do not directly kill infected cells but are capable of stifling their reproduction and boosting the proliferation of effector cells. They can also act in a suppressive manner and hinder effector cell proliferation, in turn boosting any nearby healthy cells. This is roughly equivalent to the behavior of T cells in our body. Effector CD8+ T-cells can effectively kill pathogens and cancerous cells. Helper CD4+ T-cells help recruit other immune cells to an inflamed site, while diminishing immune responses and helping nearby cells in healthy tissue [11].

Modeling

We modeled the behavior of immune cells as a Markov Decision Process, with each individual immune cell attempting to maximize its own utility with respect to its local environment. An immune cell can take one of two actions: it can either traverse the grid, or attack its neighboring cells, killing all immediately adjacent cells indiscriminately if successful. We designed two different utility functions to help our immune cells navigate the simulation with the goal of killing infected cells.

The first utility function we created was aptly titled NaiveUtility. The NaiveUtility function, in the case of a cell attacking, rewards attacking infected cells and punishes attacking healthy cells equally. The counts of healthy and infected cells attacked are multiplied by s, which here represents the probability of an attack successfully going off. In the case of a move, it returns .5 times the number of possible empty cells around it that are available to move into. When moving, it randomly selects its destination from its list of empty neighbors.

NaiveUtility will prioritize attacking if there are enough neighboring infected cells around it to attack. It should be noted however that it does not have any encoded logic for how to move, and weights attacking immune cells and saving healthy cells with the same priority. We decided to build on NaiveUtility and create a utility function that would better encode the behavior of actual immune cells, which target and hunt down infections regardless of the collateral damage they might cause. To this end, we created SmartUtility.

$$SmartUtility(c,a) = \begin{cases} 2.5s * n_{infected} - 1.5s * n_{healthy} - .5s * n_{immune} + .03 * n^* & a = attack \\ .8/d & a = move \end{cases}$$

When attacking, SmartUtility more heavily rewards killing healthy cells than it punishes attacking healthy or immune cells. It also accounts for nearby immune cells that attacked in the current time step, represented by n*. When the move action is chosen, SmartUtility multiplies a constant by the inverse distance to the nearest infected cell, represented by d. In this way SmartUtility attempts to reward moving closer to infected cells without killing unnecessary healthy and immune cells, unless there is an infected cell available to attack.

We also added an additional type of Immune cell, a helper cell, which behaves differently from the base immune cell type. These helpers cannot attack infected cells, but have the ability to suppress their reproduction rate, and can boost the reproduction rates of healthy and immune cells. These helpers have their own utility function:

$$HelperUtility(c,a) = \begin{cases} n_{infected} * \frac{n}{r^2} & a = suppress \\ n_{healthy} * \frac{n}{r^2} & a = boost \\ .01/d & a = move \end{cases}$$

HelperUtility determines whether to suppress or boost local cells by doing density calculations over their local environment within the grid (the radius of this area denoted by r). It compares the number of healthy or infected cells within the local area to the density of cells in that local area and tries to move into more densely packed regions of the environment if possible.

Simulation

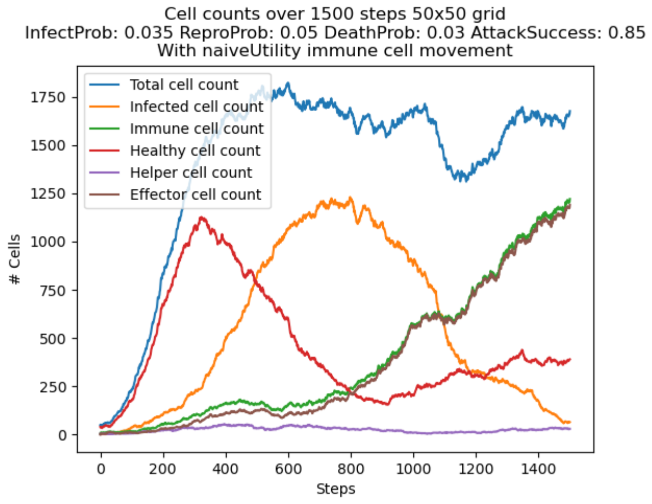
We represented the population of healthy, infected, and immune cells in a two-dimensional array, with each row, column pair corresponding to a position in the grid. At initialization, healthy, infected, and immune cells are randomly populated through the grid. With each time step through our simulation, cells have a chance to reproduce, become infected/infect neighboring cells, and in the case of immune cells, take an action (which are chosen by the immune cell's utility function). At the end of each time step, each cell has a probability of dying.

We controlled the simulation by tuning a number of hyperparameters, including the reproduction, infection, and death probabilities, the attack success constant, the helper boost constant, as well as the grid size and initial cell population counts.

Data

To quantify our agents' success, we kept track of a multitude of global parameters over the course of our simulations. These parameters can be grouped into two categories: cell counts, and cell actions. The overall goal of the data is to measure what the state of the world is temporally as the agents move in their stochastic environment. Cell counts include: all cells, healthy cells, infected cells, immune cells, and subtypes of immune cells (effector and healthy). Cell actions, meanwhile, are: reproducing, moving, infecting, dying, activating (immune cells), killing (effector immune cells), boosting and suppressing (both helper immune cell actions).

Results & Discussion



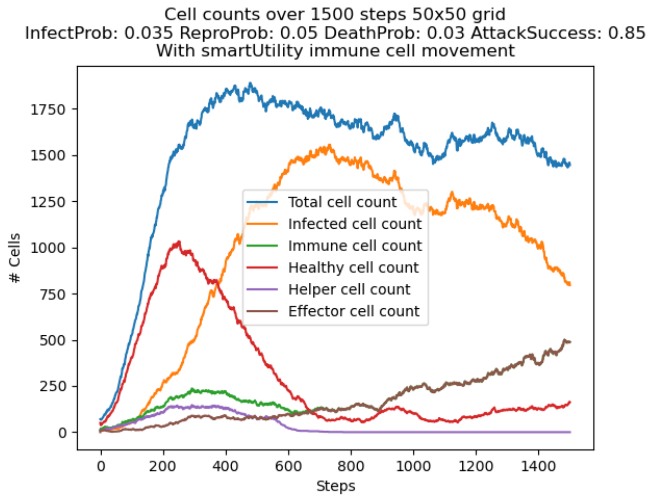


Figure 1: Cell counts when immune cells use the NaiveUtility function (left) or the SmartUtility function (right).

From our simulation, the SmartUtility function does not meaningfully increase the cell counts of healthy cells in the population and can at times be detrimental to the healthy cell population. Not shown are the terminal outputs of the grid as the game progresses. While the NaiveUtility immune cells proceed with a random walk, the SmartUtility cells should try and move towards infected cells. However, we found that many of the cells will move close to infected cells before stopping several squares away outside of attacking range and just sit there. This indicates some issues with our utility function preferring attacking to moving before expected, but we were unable to locate exactly why. The NaiveUtility function does a better job at controlling infected cell population but is unable to return the healthy population to a majority of cells, instead electing to maximize immune cell numbers.

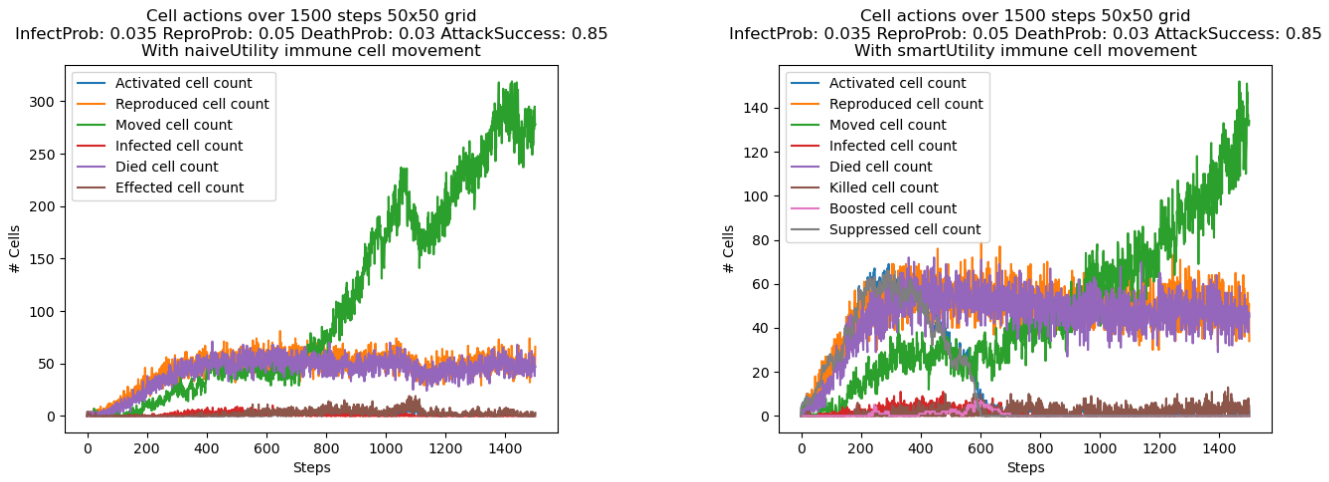


Figure 2: Cell actions when immune cells use the NaiveUtility function (left) or the SmartUtility function (right).

Looking now at the cell actions during the same simulation, we can see the NaiveUtility cells prefer the move particularly in the end game when they vastly outnumber infected cells. The SmartUtility immune cells also have a preference for moving, although not to the same extent as the NaiveUtility cells, though this could be due to their lower cell count in this simulation.

An important constraint to these simulations is their computational complexity. A single run takes over an hour for each utility function, likely due to its lack of computational efficiency and being called for every immune cell on every step. Optimizations are needed to improve performance and allow scalability. Additionally, as the environment and many behaviors are stochastic, it would be best to look at several simulations with the same conditions and plot the average and standard deviation of each count. Attempting to do this on with our current code led to train times beyond 24 hours and was outside the scope of this project.

While our project was successful in implementing all the steps of our grade contract in a unified Python game, the outcome from these simulations was not what we sought for[1]. Ideally, our simulation would mimic the human immune response, where immune cells make up a tiny fraction of our total cells but rapidly expand in the local areas of inflammation and infection before self-correcting their population. Other groups have successfully deployed agent-based models which recapitulate this functionality and will be closely read to understand how they explain these results using their immune cell utility (Figure 3).

[1] Our project implementation is available here: https://github.com/hayitsian/Agent-Based-Immune-Modeling

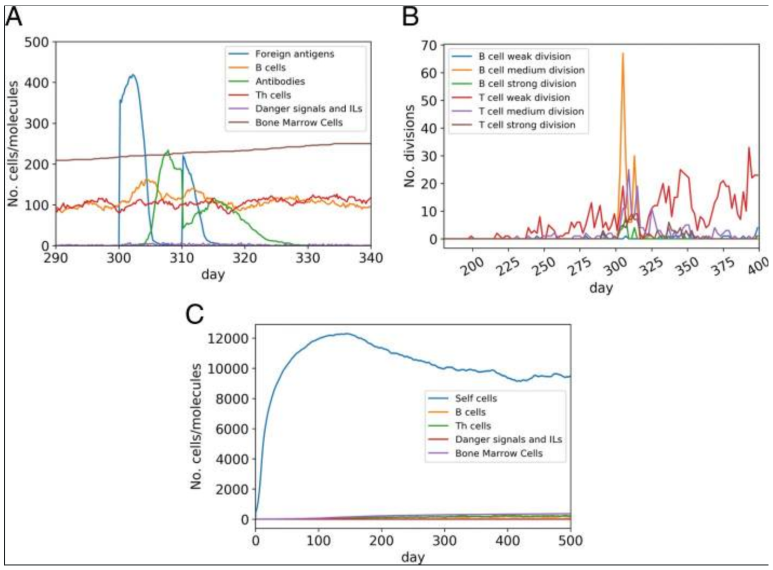


Figure 3: Cell count results from MiStImm, an agent-based model of the human immune system with distinct cell types (B cells, Th cells), important molecules (Antibodies), weak & strong cell division, and multiple infection types (cancer, bacteria, viruses, etc.) [9].

Future Directions

In the realm of modeling the human immune system, there are countless ways to more accurately recreate perhaps the most complicated organ system in our bodies. There are dozens of immune cell subtypes, each with certain functions and features that could be explored in agent-based models. Cellular communication is far more complicated than a binary output in a fixed local area; some groups have already experimented by defining proteins such as interleukins or cytokines, and their appropriate receptors on some (but not all) cell types. Additionally, there are antibodies, perhaps the most important protein in our immune response that guides our immune cells and is a key component of immune memory. Another component of the immune system is its two facets: adaptive and innate. Innate immunity refers to the non-specific killing of anything considered foreign to the body: viruses, bacteria, even cancerous cells. Adaptive immunity develops specific target proteins on the invader, called antigens, that it uses to guide its targets and eliminate a particular foe. Finally, the immune system is built around a component self- versus non-self, which it uses to guide both innate and adaptive immunity.

On a smaller scale, there are also some immediate improvements to improve the accuracy of our model. First is the inclusion of spontaneous infections. Currently infected cells are only defined at the initial timestep; once the infected cells are gone, there is nothing for the immune cells to do other than try and boost healthy cell reproduction through helper cell signaling. Additionally, we are only considering a single type of infection here: viral. Viral infections overcome human cells and use them to create more viruses, spreading to nearby cells. Bacterial infections are fundamentally their own cells that do not spread to human cells but rather compete with them for resources (but, in the case of the intestinal tract, are also essential to healthy function). Finally, there are cancerous cells, which are in a way the same identity as healthy cells but mutated to outcompete local healthy cells for resources. Each of these can and has been implemented in similar-scale models by research groups studying immunological responses and would be a logical next-step for this project.

Works Cited

[1] Z. Lin *et al.*, “Language models of protein sequences at the scale of evolution enable accurate structure prediction.” bioRxiv, p. 2022.07.20.500902, Jul. 21, 2022. doi: 10.1101/2022.07.20.500902.

[2] J. Jumper *et al.*, “Highly accurate protein structure prediction with AlphaFold,” *Nature*, vol. 596, no. 7873, Art. no. 7873, Aug. 2021, doi: 10.1038/s41586-021-03819-2.

[3] “Quantitative Fundamentals of Molecular and Cellular Bioengineering,” *MIT Press*. https://mitpress.mit.edu/9780262042659/quantitative-fundamentals-of-molecular-and-cellular-bioengineering/ (accessed Apr. 25, 2023).

[4] A. Handel, N. L. La Gruta, and P. G. Thomas, “Simulation modelling for immunologists,” *Nat. Rev. Immunol.*, vol. 20, no. 3, Art. no. 3, Mar. 2020, doi: 10.1038/s41577-019-0235-3.

[5] C. R. B. Bonin, G. C. Fernandes, R. W. Dos Santos, and M. Lobosco, “Mathematical modeling based on ordinary differential equations: A promising approach to vaccinology,” *Hum. Vaccines Immunother.*, vol. 13, no. 2, pp. 484–489, Feb. 2017, doi: 10.1080/21645515.2017.1264774.

[6] P. S. Kim, D. Levy, and P. P. Lee, “Modeling and Simulation of the Immune System as a Self-Regulating Network,” in *Methods in Enzymology*, Elsevier, 2009, pp. 79–109. doi: 10.1016/S0076-6879(09)67004-X.

[7] C. Cockrell and G. An, “Sepsis reconsidered: Identifying novel metrics for behavioral landscape characterization with a high-performance computing implementation of an agent-based model,” *J. Theor. Biol.*, vol. 430, pp. 157–168, Oct. 2017, doi: 10.1016/j.jtbi.2017.07.016.

[8] X. Tong, J. Chen, H. Miao, T. Li, and L. Zhang, “Development of an Agent-Based Model (ABM) to Simulate the Immune System and Integration of a Regression Method to Estimate the Key ABM Parameters by Fitting the Experimental Data,” *PloS One*, vol. 10, no. 11, p. e0141295, 2015, doi: 10.1371/journal.pone.0141295.

[9] C. Kerepesi, T. Bakács, and T. Szabados, “MiStImm: an agent-based simulation tool to study the self-nonself discrimination of the adaptive immune response,” *Theor. Biol. Med. Model.*, vol. 16, no. 1, p. 9, May 2019, doi: 10.1186/s12976-019-0105-5.

[10] V. A. Folcik, G. C. An, and C. G. Orosz, “The Basic Immune Simulator: An agent-based model to study the interactions between innate and adaptive immunity,” *Theor. Biol. Med. Model.*, vol. 4, p. 39, Sep. 2007, doi: 10.1186/1742-4682-4-39.

[11] Y. Xiong and R. Bosselut, “CD4-CD8 differentiation in the thymus: connecting circuits and building memories,” *Curr. Opin. Immunol.*, vol. 24, no. 2, pp. 139–145, Apr. 2012, doi: 10.1016/j.coi.2012.02.002.