Parallelization of a stochastic model for exploring the evolutionary dynamics of antibiotic resistance

Lauren Corbin
Duke University
Durham, North Carolina
lauren.corbin@duke.edu

Helena Ma
Duke University
Durham, North Carolina
helena.ma@duke.edu

ABSTRACT

Understanding the evolutionary dynamics of antibiotic resistance in heterogeneous bacterial populations is essential to designing treatments and policy to minimize the spread of resistance. These dynamics can be understood as a complex interaction between the differential growth rates and differential protection from antibiotic-induced lysis for different bacterial subpopulations, which also depends on spatial structure of bacterial populations and antibiotic exposure. In this work, we develop a stochastic cellular automata model that accounts for basic processes of antibiotic diffusion and degradation and cellular death, division, and subpopulation switching. To simulate large bacterial populations of over 10^7 cells, we implement this model in parallel and introduce optimizations to improve parallel performance. Experiments demonstrate a maximum speedup of 57× the serial implementation. Using our model, we demonstrate cellular response to the presence of antibiotic, providing a proof-of-principle for use of this model for scientific exploration of antibiotic resistance dynamics.

CCS CONCEPTS

• **Computing methodologies** → *Massively parallel algorithms*; *Agent / discrete models*; • **Applied computing** → *Systems biology*.

KEYWORDS

antibiotic resistance, agent-based models, cellular automata

1 INTRODUCTION

Antibiotics are a cornerstone of modern medicine which have not only drastically reduced mortality caused by deadly bacterial diseases but also protect patients in any vulnerable immune state, including surgery patients, transplant recipients, AIDS patients, and more. Unfortunately, bacteria have evolved many resistance mechanisms that allow them to survive antibiotic treatment. Many of these medicines are now useless against the diseases they are intended to treat, and due to disincentivizing economics, the antibiotic pipeline is thin [9].

It is thought that frequent exposure caused by overprescribing has accelerated this rise of antibiotic resistance, giving a greater survival advantage to strongly resistant bacteria [4]. However, bacterial populations are not typically homogeneous, consisting instead of a mixture of sensitive and resistant bacteria. When the mechanism of resistance is one that degrades antibiotic in the environment, the production of this mechanism can be thought of as a public good, with resistant cells cooperating to benefit the entire population. The presence of antibiotic selects for this cooperation. However, in some cases, if antibiotic concentrations are nonlethal and low, sensitive bacteria may also survive by taking advantage of the resistance produced by the cooperators. These cheaters, which do not bear the metabolic burden of producing resistance, may in some cases actually dominate the population [10]. A better understanding of how large heterogeneous populations respond to antibiotic treatment, including which conditions most strongly favor the proliferation of resistance, will be essential to deploying antibiotics in a way that can prolong their use as well as designing new antibiotics that are less prone to resistance development.

One way to understand these dynamics is to use evolutionary game theory [2]. Individual cells can be thought of as players, and their genotypes (for instance, cheater or cooperator) can be thought of as strategies. Their fitness depends on the strategy of individual cells and their neighbors. By simulating games, it is possible to determine which strategy is the most fit for a given set of environmental conditions. Previous work has modeled populations of cooperators and cheaters using deterministic models of partial differential equations [11]. These models are good for modeling the behavior of cells in a homogenous, well-mixed environment. However, previous studies, both experimental [1] and *in silico* [3, 6], have also shown that spatial structure can be influential in determining the outcomes of such games. Despite this, simulations that account for spatial structure to date are limited in flexibility, frequently assuming constant antibiotic concentrations and randomized initialization of cells.

In this paper, we first develop a stochastic model that can explore evolutionary dynamics of resistant and sensitive bacterial cells in response to antibiotics while accounting for spatial effects. Our model accounts for antibiotic degradation and diffusion by cells and is generalizable to very large communities. Colonies of *Escherichia coli* may contain $10^7 - 10^8$ cells within 12 hours [7], suggesting that models must be at large scale to model more physiological spatial effects; previous models are restricted to at most 62,500 cells [6]. Furthermore, if the temporal resolution is low, simulating relevant timespans, even on small populations, may also become computationally expensive. In order to achieve good performance at large lattice sizes and for long simulation times, we parallelize the model using the Message Passing Interface (MPI) and study its scalability.

2 MODEL IMPLEMENTATION

The model we developed is an extension of cellular automata such as the original Game of Life [5], in which initial conditions are set and cells act according to their state. We represent cell type C and antibiotic concentration [A] over a regular lattice using two-dimensional matrices. Cells may be resistant (R) or sensitive (S). *Neighbors* of a cell in the lattice are considered all those with x and y

Table 1: Specifiable Parameters

Process	Parameter	Value	Significance
Diffusion	k_{diff}	0.1	Rate of antibiotic diffusion
Degradation	k_{deg}	1	Rate of antibiotic degradation by resistant cells
Death	MBC_r	400	Minimum bactericidal concentration of resistant cells
Death	MBC_s	4	Minimum bactericidal concentration of sensitive cells
Death	P_{death}	0.5	Probability of death if $[A] <$ MBC
Division	k_{div_s}	$\frac{1}{1800}$	Normal rate of division
Division	β	0.25	Resistance penalty, where the resistant rate of division $k_{div_r} = \beta\, k_{div_s}$
Switching	k_{switch}	$\frac{1}{1800}$	Rate of cell type switching

coordinates within 1 of those of the cell, for a total of 8 neighbors per cell. Initial concentrations and cell types are initialized randomly, according to a predefined pattern, or read in as input to the program. Time advances in steps of $\Delta t = 0.25$ seconds. At each timestep, the following processes occur. A summary of adjustable parameters that govern these processes can be found in Table 1.

2.0.1 Antibiotic Diffusion. Antibiotic diffusion is implemented in two dimensions according to the diffusion equation:

$$\frac{\partial \phi}{\partial t} = k_{diff} \frac{\partial^2 \phi}{\partial x^2}$$

For each lattice point, changes in antibiotic concentration due to diffusion into the cell are dependent on the concentration in adjacent lattice points, in the x and y dimensions, at the previous timestep. Changes in concentration due to diffusion away from the cell are dependent on the concentration of the cell at the previous timestep.

2.0.2 Antibiotic Degradation. Antibiotic can be degraded by resistant cells only. Antibiotic degradation is implemented using first-order kinetics:

$$\frac{\partial [A]}{\partial t} = -k_{deg}[A]$$

2.0.3 Cell Death. We implement cell death stochastically depending on antibiotic concentration and cell type. For bacterial cells to be killed by an antibiotic, its concentration must be greater than the minimum bactericidal concentration (MBC). We assume that $MBC_r \geq MBC_s$. In this model, a given cell has a probability P_{death} of dying if [A] > MBC.

2.0.4 Cell Division. We implement cell division stochastically depending on cell type. We define a division rate for sensitive cells k_{div_s}. We assume that resistance carries a penalty β, which is a real number between 0 and 1, where the division rate for resistant cells $k_{div_r} = \beta k_{div_s}$. The locations of empty neighbors (locations with no cells) are tabulated, and the cell will produce another cell of the same type in a location randomly chosen among these empty neighbors with probability k_{div}.

2.0.5 Cell Type Switching. We implement cell type switching stochastically. At any step, cells may switch from one type to another with a probability P_{switch}, simulating the possibility of cells gaining or losing resistance via mutations or horizontal gene transfer.

3 PARALLEL OPTIMIZATIONS

In order to run simulations in parallel, we used MPI to run the process on multiple nodes. This necessitated a number of optimizations to the code.

3.1 Master-Worker Architecture

Because output generation was found to be a major performance bottleneck in our initial model development, and because output is generated multiple times throughout the simulation, when adding parallelization we set up a master-worker architecture so that the master rank does not handle any part of the lattice and is the only node responsible for outputting data (Fig. 1). This allows simulations for further timesteps to proceed while output is being generated.[1] To further improve the output speed, we used binary output using MPI_File_write().

3.2 Domain Decomposition and Halo Exchanges

We decomposed the domain in one dimension along the Y axis (Fig. 1). Each node handles a subset of rows in the lattice, with the number of rows per node being evenly distributed for load balancing purposes. Since cell division and antibiotic diffusion processes are dependent on [A] and C of neighboring lattice points, nodes communicate single-row halos of their first and last rows to the previous and following nodes, respectively. This occurs at the beginning of each timestep. In order to reduce memory usage per node, each rank keeps small [A] and C arrays of only its rows and halos, such that only the master node stores arrays of the full lattice size.

3.3 Process Decoupling

In the original model implementation, the five processes ran separately, such that at every timestep diffusion would be calculated for the entire lattice, followed by degradation for the entire lattice, and so and and so forth for cellular death, division, and switching. However, implementing this in parallel would require halo exchanges after each process, requiring five times as many synchronizations between nodes. In order to reduce intranode communication, we

[1] Since the master rank does not handle any part of the lattice, in our scaling experiments we plot only the number of workers n, although the model uses $n + 1$ nodes.

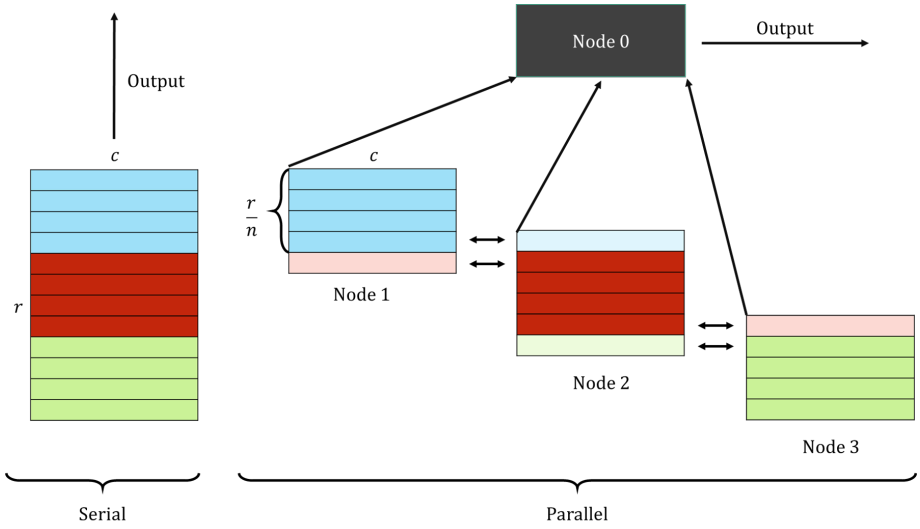

Figure 1: Depiction of serial and parallel architecture and domain decomposition for an $r \times c$ lattice where each node in the parallel architecture handles $\frac{r}{n}$ rows. Halo exchanges occur between adjacent nodes while the master node handles output.

decoupled these processes by rewriting functions to return values for a single cell instead of updating the entire lattice and making all functions dependent on the previous time step.

There are three principal concerns with this redefinition of functions. The first is memory cost: in order to reduce communication overhead, we accept a tradeoff of increased memory, since each node must store [A] and C for both the current and previous timesteps. Since in a massively parallel simulation the number of rows handled by each node is greatly reduced with respect to a serial implementation, we decided that this tradeoff was worth the benefit to speedup.

The second is a difference in modeling accuracy, in which, for instance, where cell death was previously based on antibiotic degradation *since* the previous timestep, with decoupling it is based solely on antibiotic concentration *at* the previous timestep. We justified this difference in results by noting that due to the short length of Δt, the difference in [A] between the previous and current timesteps is minute.

Finally, since empty neighbors are tabulated according to the previous timestep, it is possible that there are collisions in division, in which two cells try to divide into the same place. In that case, our implementation is biased such that the cell with greater x and y index will overwrite the other cell. However, again because of the short length of Δt, the probability that two adjacent cells divide during a timestep is less than $\frac{1}{1000}$, with an even lower probability that they attempt to divide into the same location. In order to avoid needing to choose winners randomly, which would require additional memory and random number generation costs, we accepted this probability. As seen in Section 4, the results for 1 node and multiple nodes overlap, suggesting that any conflicts generated are not large.

3.4 Random Number Generation

Since random number generation is implemented in C as an atomic operation and all cellular processes are implemented stochastically and require frequent random number generation, the addition of nodes for parallelization based solely on the improvements described above did not produce speedup without parallelization of random number generation as well. To produce scalable random number generation, we used SPRNG Version 2.0 [8], which allowed us to achieve speedup.

4 VALIDATION

Because our model implementation is stochastic, in order to validate the performance of our improved parallel implementation, we monitored the amount of resistant and susceptible cells over time over three independent runs with the same parameter set for increasing numbers of nodes (Fig 2). The results are not significantly different from one another, suggesting that the parallel implementation gives similar results to a serial implementation.

5 RESULTS

5.1 Computing Platform

All experiments in this study were conducted on Stampede2 at the Texas Advanced Computing Center at the University of Texas at Austin. Stampede2 hosts 4200 Intel Xeon Phi 7250 (Knights Landing) compute nodes running at 1.4 GHz, which include 68 cores with 96 GB RAM, 16 GB MCDRAM, 32 KB L1 cache and 1 MB L2 cache per two cores. The programs are written in C and use MPI for parallelism.

5.2 Performance Results

5.2.1 Strong Scaling. For our strong scaling experiments, we first measured the runtime of a 5000×5000 lattice executing for 30 seconds (Fig. 3). We achieve a max speedup of 22.2× our single-node

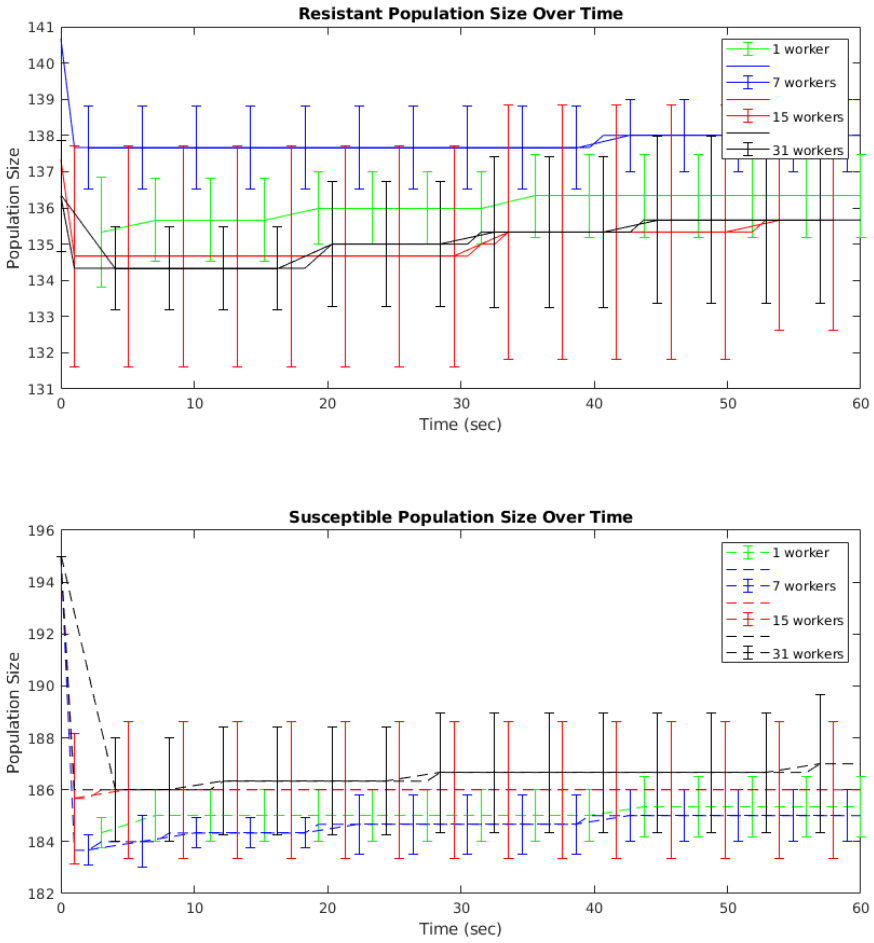

Figure 2: Average and standard deviation of numbers of resistant and susceptible cells over time for three independent runs of varying nodes.

implementation. This is noticeably below 64×, but still significant speedup.

We then measure the runtime of a 1000 × 1000 lattice executing for 3600 seconds (Fig. 4). We achieve a max speedup of 57.8× our single-node implementation, which is very close to 64×. The discrepancy between the scaling for the long simulation and the large simulation may be due to the memory demands. We note that the 32Kb L1 cache may only hold approximately 8,000 4-byte floats at one time, suggesting that with sufficiently high numbers of columns, if the cache is loaded by row, accessing the 8 neighbors may require frequent evictions of the cache.

5.2.2 Weak Scaling. For our weak scaling experiments, we performed three experiments (Fig. 5). For each, we increased the number of rows, columns, or timesteps proportionally with the number of nodes. Weak scaling results were irregular before approximately ten nodes. We find that weak scaling efficiency on 63 workers as compared to 9 workers was 105% for rows, 82% for columns, and 45% for time. We note that increasing the number of rows increases the number of elements belonging to each node, but does not increase the size of halos communicated. However, increasing the number of columns increases both the number of elements and the size of halos communicated. This suggests that communication is an important bottleneck, since weak scaling efficiency was better when only elements processed increased, as opposed to elements processed. Increasing time should proportionally increase both time spent processing and time spent communicating, which may explain why scaling efficiency was worst in this dimension.

5.2.3 Performance Bottlenecks. In order to evaluate performance bottlenecks, we have constructed a simple performance model for a lattice of r rows and c columns. For each timestep, the processes

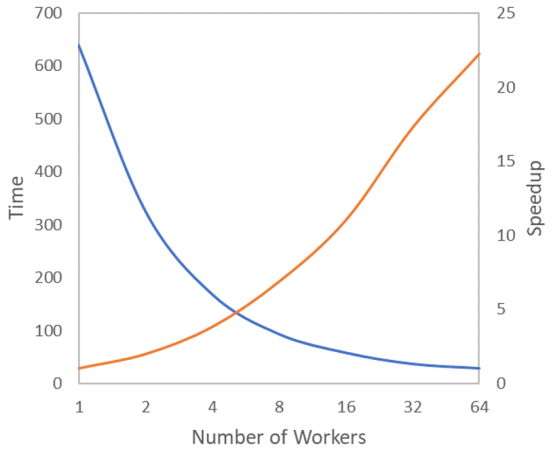

Figure 3: Strong scaling with speedup and runtime on a 5000 × 5000 board for 30 seconds.

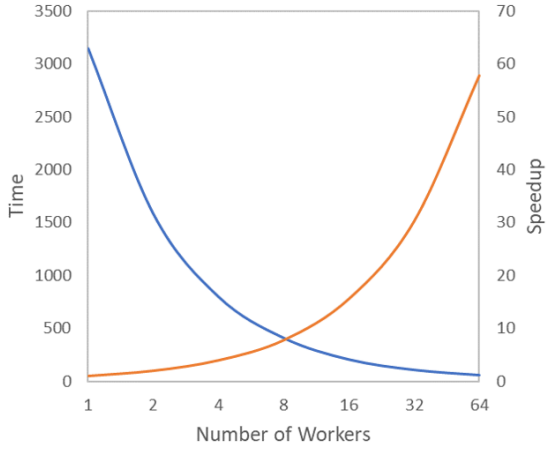

Figure 4: Strong scaling with speedup and runtime on a 1000 × 1000 board for 3600 seconds.

can be broken down as in Table 2, with the total time spent running each process found by multiplying the number of timesteps T by $c \times \frac{r}{n}$. Neighbor counting, a subprocess of cell division, is the most costly process for all operations except random number generation; however, the most costly process in RNG is also cell division, suggesting that the cell division process is the largest bottleneck among the five processes described in Section 2.

Halo exchange, as modeled in Table 3, requires significantly fewer memory accesses and comparisons but requires inter-node communication. Cell division will still take more time than halo exchanges unless the time it takes to send one entry is greater than $(\frac{103r}{4nc} - \frac{4}{c})M + \frac{4r}{n}F + (\frac{16r}{n} - \frac{11}{c})I + \frac{3R}{4c}$, where M is the time for one memory access, F is the time for one floating point operation, I is the time for one comparison, and R is the time for one random number generation. Communication can also become costly in the

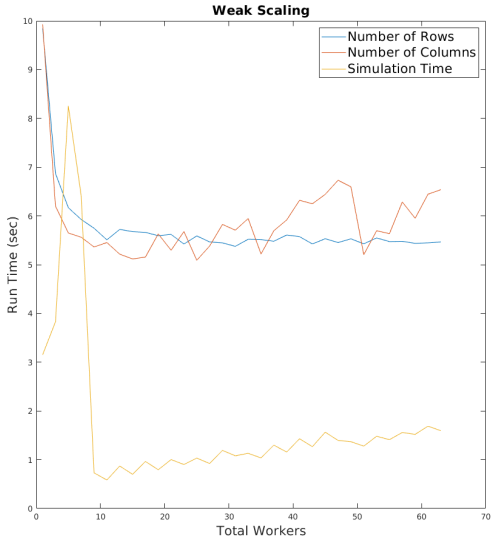

Figure 5: Weak scaling on number of rows, number of columns, and number of timesteps. Constants for number of columns: 100 rows, 600 timesteps. For number of columns: 100 columns, 600 timesteps. For number of timesteps: 100 rows, 100 columns.

event of frequent data outputting, depending on T_{out}, the interval of timesteps after which the model produces output. We can see in our weak scaling results that communication size, from number of columns, does have a significant impact on the runtime, since weak scaling efficiency was worse when the number of columns increased but the output interval stayed the same.

Finally, the time to generate output is modeled in Table 4. This is executed at the same time as computation and communication for each timestep. This means that whichever is longer among Tables 2 and 3 and Table 4 will be the bottleneck, depending primarily on the length of T_{out}.

In order to test these predictions experimentally, we computed runtimes for each of these processes for a simulation of a 1000 × 1000 lattice on 4 nodes for 10 seconds, producing output once every ten seconds, generating in timings found in Table 5. As shown in Figure 6, out of the five per-point processes our model does indeed spend the most time computing cell division. Figure 6 also shows that, as we predicted, with frequent output, output may rival cell division for the most costly process, while if output is less frequent then cell division and other processes will dominate.

5.3 Scientific Results

A thorough exploration of the impact of the parameters in Table 1 was outside the scope of this paper. However, we present here some basic results for the population dynamics to demonstrate the utility of this model. Figures 7 and 8 show the evolution of cell type and antibiotic concentration of a 100 × 100 lattice of horizontal stripes of bacteria juxtaposed against vertical stripes of antibiotic. We

Table 2: Performance Model - Processes - Frequency: T × c × $\frac{r}{n}$

Process	Loads	Stores	Floating Point Operations	Comparisons	RNG
Copy Matrices	2	2	-	-	-
Diffusion	11	1	13	-	-
Degradation	5	2	2	1	-
Death	8	5	-	4	1
Division	11	3	-	8	**3**
Count neighbors	**51**	**38**	**16**	**56**	-
Switching	6	2	-	4	1

Table 3: Performance Model - Communication

Process	Loads	Stores	Comparisons	Communication	Frequency
Halo Exchange	$8c$	$8c$	11	$4c$	T
Pre-output Gather	$2cr$	$2cr$	1	$2c\frac{r}{n}$	$\frac{T}{T_{out}}$

Table 4: Performance Model - Output - Frequency: $\frac{T}{T_{out}}$

Process	Loads	Stores	Comparisons	File Operations	Write
Pre-output Gather	$2cr$	-	-	-	-
Output	$2cr$	$2cr$	cr	4	$2cr$

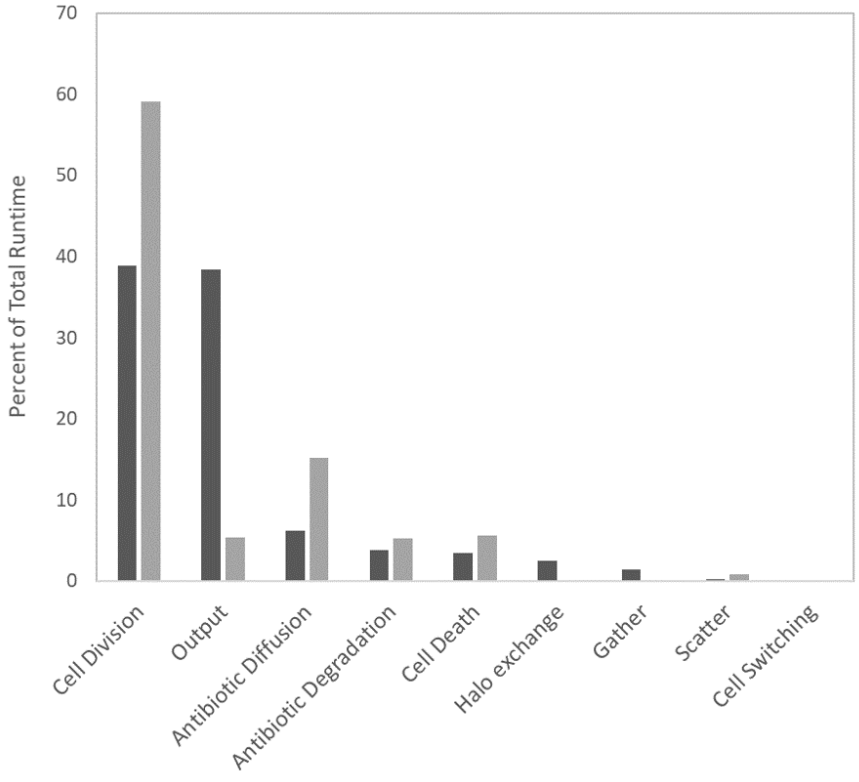

Figure 6: Most costly operations in order of percentage of total runtime for a 1000 X 1000 matrix outputting once every 4 steps (dark gray) and once every 20 steps (light gray).

Table 5: Timings for Individual Computations

Process	Time (sec)	Percent of Total Time
Diffusion	0.570486	6.2252
Degradation	0.351693	3.8377
Cell switching	0.001725	0.0188
Cell death	0.322045	3.5142
Cell division	3.558697	38.8326
Halo exchange	0.229358	2.5028
Scatter	0.023713	0.2588
Gather	0.133087	1.4522
Output	3.512378	38.3272
Total time	9.164198	-

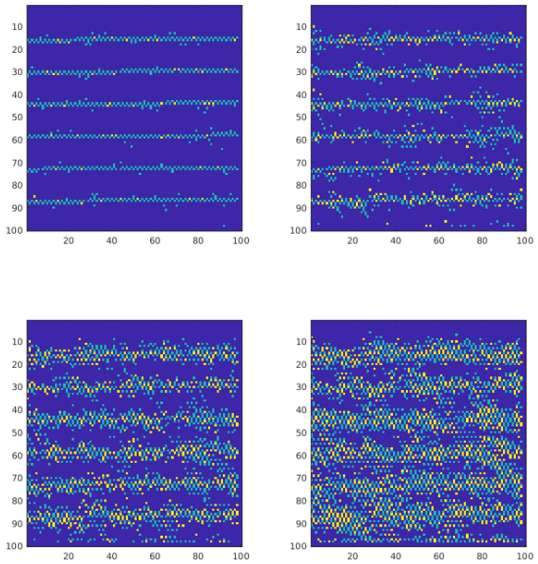

Figure 7: Pictorial representation of bacterial populations over time; yellow is resistant, turquoise is sensitive

observe that bacteria aggregate toward a spatial structure, and that resistant cells appear to be more prominent near positive antibiotic concentrations. Figure 9 shows that in this regime, resistant cells are selected for, with the proportion of resistant cells increasing over time.

6 CONCLUSIONS

In this work, we developed a stochastic model of evolutionary dynamics of antibiotic resistance which incorporates spatial effects from antibiotic diffusion and degradation as well as cellular processes of death, division, and transitions between sensitivity and resistance. We implemented this model in parallel, achieving speedup of 57.8× over our initial serial implementation for a long

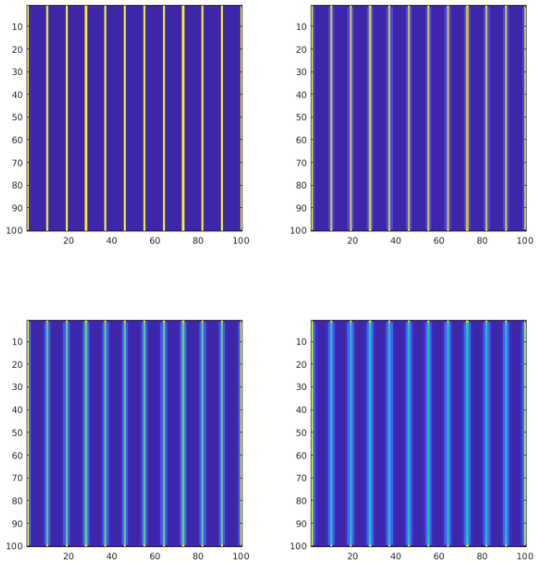

Figure 8: Pictorial representation of antibiotic concentration over time; yellow is high antibiotic, light blue is lower antibiotic

count.png

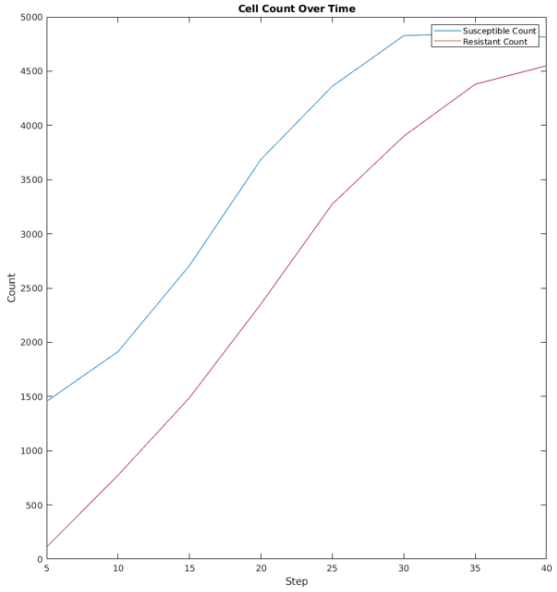

Figure 9: Cell counts of sensitive (blue) and resistant (red) cells over time

simulation time and 22.2× for a large simulation. We identified that further optimization should focus on the cell division process, which is by far the most costly operation, especially if output is infrequent. We then used this model with some default parameters to demonstrate stochastic proliferation of a bacterial population and a regime under which resistant cells are selected for in the presence of antibiotics, suggesting that further parameter exploration may help to investigate antibiotic resistance dynamics.

Our model definition is very flexible and can be easily extended to test the influence of different parameters on the proliferation of bacteria and of resistance. In particular, during future work the importance of the following can be evaluated without significant structural change to our model:

- patterns and concentration of initial antibiotic
- periodic dosing
- patterns of initial cell type
- antibiotic degradation rate
- differential MBC between populations
- magnitude of resistance burden
- frequency of cell type switching

More complex forms of regulation, such as a dependence of cell type switching and other parameters on the identity of neighboring cells, could also be implemented with some alterations to the process functions, using our neighbor counting function. Additionally, with some alterations to data representation, the model could be augmented to account for a greater range of cell types. This flexibility suggests that this model may be used as an efficient tool for the scientific investigation of antibiotic resistance evolutionary dynamics in heterogeneous populations.

Finally, we note that a limitation to our model is its implementation in two dimensions, while true bacterial growth on surfaces occurs in colonies or biofilms which exhibit three-dimensional structure. Extending the model to three dimensions and integrating some concepts from study of bacterial structures to closer approximate physiological realities is an interesting future direction.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Amanda Randles and Daniel Puleri for support and guidance over the duration of this project. This work used resources of the Texas Advanced Computing Center (TACC).

REFERENCES

[1] Lin Chao and Bruce R. Levin. 1981. Structured habitats and the evolution of anticompetitor toxins in bacteria. *Proceedings of the National Academy of Sciences* 78, 10 (Oct. 1981), 6324–6328.

[2] Peter L. Conlin, Josephine R. Chandler, and Benjamin Kerr. 2014. Games of life and death: antibiotic resistance and production through the lens of evolutionary game theory. *Current Opinion in Microbiology* 21 (Sept. 2014), 35–44. https://doi.org/10.1016/j.mib.2014.09.004

[3] Rick Durrett and Simon Levin. 1997. Allelopathy in Spatially Distributed Populations. *Journal of Theoretical Biology* 185 (Sept. 1997), 165–171. https://doi.org/10.1006/jtbi.1996.0292

[4] Centers for Disease Control and Prevention. 2013. Antibiotic Resistance Threats in the United States. (April 2013).

[5] Martin Gardner. 1970. Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American* 223 (Oct. 1970), 120–123.

[6] Benjamin Kerr, Margaret A. Riley, Marcus W. Feldman, and Brendan J. M. Bohannan. 2002. Local dispersal promotes biodiversity in a real-life game of rock paper scissors. *Nature* 418 (July 2002), 171–174. https://doi.org/10.1038/nature00823

[7] Harvey Lodish, Arnold Berk, S. Lawrence Zipursky, Paul Matsudaira, David Baltimore, and James Darnell. 2000. *Molecular Cell Biology - Manipulating Cells and Viruses in Culture.* W.H. Freeman, New York, NY, USA, Chapter 6, 51–74. https://www.ncbi.nlm.nih.gov/books/NBK21475/

[8] Michael Mascagni and Ashok Srinivasan. 2000. Algorithm 806: SPRNG: a scalable library for pseudorandom number generation. *ACM Trans. Math. Software* 26, 3 (Sept. 2000), 436–461. https://doi.org/10.1145/358407.358427

[9] John H. Rex, Barry I. Eisenstein, Jeff Alder, Mark Goldberger, Robert Meyer, Aaron Dane, Ian Friedland, Charles Knirsch, Wendy R. Sanhai, John Tomayko, Cindy Lancaster, and Jennifer Jackson. 2013. A comprehensive regulatory framework to address the unmet need for new antibacterial treatments. *The Lancet Infectious Diseases* 13, 3 (March 2013), 269–275. https://doi.org/10.1016/S1473-3099(12)70293-1

[10] Michael Travisano and Gregory J. Velicer. 2004. Strategies of Microbial Cheater Control. *TRENDS in Microbiology* 12, 2 (Feb. 2004), 72–78. https://doi.org/10.1016/j.tim.2003.12.009

[11] Eugene A. Yurtsev, Hui Xiao Chao, Manoshi S. Datta, Tatiana Artemova, and Jeff Gore. 2013. Bacterial cheating drives the population dynamics of cooperative antibiotic resistance plasmids. *Molecular Systems Biology* 9, 683 (June 2013). https://doi.org/10.1038/msb.2013.39