5-2017

A Parallel Genetic Algorithm for Optimizing Multicellular Models Applied to Biofilm Wrinkling

Christopher Douglas Johnson
Utah State University

A PARALLEL GENETIC ALGORITHM FOR OPTIMIZING MULTICELLULAR MODELS APPLIED TO BIOFILM WRINKLING

by

Christopher Douglas Johnson

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Nicholas Flann, Ph.D. Major Professor	Vicki H. Allan, Ph.D. Committee Member
Gregory J Podgorski, Ph.D. Committee Member	Mark R. McLellan, Ph.D. Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2016

ABSTRACT

A Parallel Genetic Algorithm for
Optimizing Multicellular Models
Applied to Biofilm Wrinkling

by

Christopher Douglas Johnson, Master of Science
Utah State University, 2016

Major Professor: Nicholas Flann, Ph.D.
Department: Department of Computer Science

Multiscale computational models integrating sub-cellular, cellular, and multicellular levels can be powerful tools that help researchers replicate, understand, and predict multicellular biological phenomena. To leverage their potential, these models need correct parameter values, which specify cellular physiology and affect multicellular outcomes. This work presents a robust parameter optimization method, utilizing a parallel and distributed genetic-algorithm software package. A genetic algorithm was chosen because of its superiority in fitting complex functions for which mathematical techniques are less suited. Searching for optimal parameters proceeds by comparing the multicellular behavior of a simulated system to that of a real biological system on the basis of features extracted from each which capture high-level, emergent multicellular outcomes. The goal is to find the set of parameters which minimizes discrepancy between the two sets of features. The method is first validated by demonstrating its effectiveness on synthetic data, then it is applied to calibrating a simple mechanical model of biofilm wrinkling, a common type of morphology

observed in biofilms. Spatiotemporal convergence of cellular movement derived from experimental observations of different strains of *Bacillus subtilis* colonies is used as the basis of comparison.

(62 pages)

PUBLIC ABSTRACT

A Parallel Genetic Algorithm for
Optimizing Multicellular Models
Applied to Biofilm Wrinkling

by

Christopher Douglas Johnson, Master of Science
Utah State University, 2016

Major Professor: Nicholas Flann, Ph.D.
Department: Department of Computer Science

Multiscale computational models integrating sub-cellular, cellular, and multicellular levels can be powerful tools that help researchers replicate, understand, and predict multicellular biological phenomena. To leverage their potential, these models need correct parameter values, which specify cellular physiology and affect multicellular outcomes. This work presents a robust parameter optimization method, utilizing a parallel and distributed genetic-algorithm software package. A genetic algorithm was chosen because of its superiority in fitting complex functions for which mathematical techniques are less suited. Searching for optimal parameters proceeds by comparing the multicellular behavior of a simulated system to that of a real biological system on the basis of features extracted from each which capture high-level, emergent multicellular outcomes. The goal is to find the set of parameters which minimizes discrepancy between the two sets of features. The method is first validated by demonstrating its effectiveness on synthetic data, then it is applied to calibrating a simple mechanical model of biofilm wrinkling, a common type of morphology

observed in biofilms. Spatiotemporal convergence of cellular movement derived from experimental observations of different strains of *Bacillus subtilis* colonies is used as the basis of comparison.

To my wife Melanie,
for her unconditional support all the way through.

ACKNOWLEDGMENTS

With deep gratitude I would like to thank my mentor Professor Nicholas Flann for his wealth of knowledge and invaluable advice provided throughout my graduate education. My thanks also extends to my colleagues at Utah State University and the Institute for Systems Biology, particularly Boris Aguilar who generously offered guidance and insight throughout our research.

Chris Johnson

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Introduction

Self-regulated spatiotemporal organization such as the formation of 3D patterns is a fundamental biological process in the development of multicellular organisms [1, 2]. Biofilms are simpler than most multicellular organisms, yet they still form complex 3D patterns in the form of interconnected wrinkles [3, 4, 5, 6]. Thus, they provide a simplified means to study 3D pattern formation.

3D patterns are a multicellular property, but their formation is guided by cellular processes like growth, cell-cell signaling, death and intercellular mechanical properties [7]. Therefore, understanding pattern formation must involve a consideration of at least the cellular and multicellular levels of organization.

Multiscale computational models are a valuable tool for understanding the dynamics of biological systems that span multiple spatial and temporal scales. A sampling of these includes biofilm modeling [8, 9]; models used to discover and develop novel drug therapies [10, 11, 12] or assess their effects and risks [13]; models focused on the molecular and sub-cellular levels including metabolism [14], gene product expression [14, 15], transcription factors [16], or the dynamics of individual molecules [17]; and agent-based multicellular models [18, 19]. References [10, 11, 15, 19] provide overviews on various types of biological modeling. See also [20] for a discussion of the many challenges in multiscale modeling.

Models like the aforementioned can be used to gain understanding of, reproduce, and predict the behavior of complex biological systems. Multicellular models include input parameters that control cellular physiology, which influences the multicellular outcome of the simulation. With inaccurate parameter values, model behavior will not accurately reproduce experimentally observed behavior and any predictions made from such a model

are likely to be of little value. Model parameters can be accurately estimated through fitting to experimental data (model calibration). This is a challenging task [21], but a variety of methods have been proposed and applied. Mahoney et al. [12] used a multi-objective simulated annealing algorithm to tune their model in an effort to discover novel cancer therapies. Baker et al. [22] applied a hybrid method combining a simple hill climbing algorithm with local parameter sweeping to fit the cellular parameters of a model of retinal pigment epithelial cells. Balsa-Canto et al. [23] combined an evolutionary technique with local search to estimate the parameters of a continuous model of metabolic cellular signaling. Lillacci and Khammash [24] developed a novel parameter-calibration method using extended Kalman filtering. These represent just a small sample of search techniques applied to the parameter-estimation of biological systems; see Sun et al. [25] for a thorough overview.

This paper presents a robust parameter fitting method and applies it to optimizing an agent-based, 3D mechanical model of biofilm wrinkling. The fitting technique utilizes a parallel and distributed genetic algorithm modified from the Multiobjective Evolutionary Algorithms (MOEA) Framework [26], an evolutionary algorithm package written in Java. The developed algorithm computes the difference (error) between multicellular, spatiotemporal features extracted from experimental data and simulation output and attempts to find a simulation which minimizes this error.

In this work, the term 'feature' refers to an abstraction of some specific measurement which quantifies the relevant emergent multicellular behavior of the biological or simulated system. The features are emergent in that they arise out of the complex and stochastic interaction of cells which do not individually contain the multicellular properties. The errors represent the spatiotemporal difference between the multicellular behavior of a real biological system and a simulated system. These errors are transformed into objectives which the genetic algorithm attempts to minimize. Thus the algorithm searches for simulations that behave similarly to the observed biological system on the basis of the spatiotemporal features extracted from each. The fitting technique introduced in this work is flexible and has the potential to be applied to many multiscale models with cellular and sub-cellular

parameters and spatiotemporal features other than the wrinkle model described in this text.

The developed algorithm is capable of running any of the following specific genetic algorithms: Nondominated Sorting Genetic Algorithm II (NSGA-II) [27], Generalized Differential Evolution (GDE3) [28], Indicator-Based Evolutionary Algorithm (IBEA) [29], Pareto Envelope-based Selection Algorithm (PESA2) [30], Strength-based Evolutionary Algorithm (SPEA2) [31], and the classic Vector Evaluated Genetic Algorithm (VEGA) [32]. These were chosen because they were implemented in the MOEA Framework and the modifica-tions necessary for this work were straightforward, e.g., parallelizing their implementation to run many distributed simulations. The data presented in this paper use NSGA-II.

The rest of this thesis is organized as follows. First, I provide some background on the biology of biofilm development and an overview of how it is represented as a computational model. A review of multi-objective optimization methods is given along with their particular application to model fitting. Next details of the simulation software and specifics of the biofilm model and biological experiments are given. The results section follows, where the method is validated with synthetic data and then applied to real data. The thesis concludes with a discussion of the evaluation, strengths, and weaknesses of the method, future work, and potential impacts.

1.2 Background

1.2.1 Biology and Model

Bacterial cells often aggregate in colonies called biofilms. These colonial biofilms are initiated by a single bacterium adhering to a surface, growing, proliferating, and secreting extracellular polymeric substances (EPS) [33], which encase the bacteria in an extracellular matrix (ECM) [34]. The EPS secreted by the bacteria acts like a glue, which keeps them attached to the surface and each other [35]. The ECM provides structural integrity to the community. As the bacteria grow and divide they push against each other causing positive pressure, but the ECM provides mechanical support between the cells that resists movement and keeps the biofilm in compression [36, 37]. Some biofilms eventually form complex

wrinkled structures. The 3D wrinkles connect and form highly permeable channels in a radial network, which maximizes liquid transport throughout the biofilm. These channels facilitate the removal of waste and the transportation of nutrients and signally molecules [38, 39]

Asally et al. [3] demonstrated that wrinkle formation is caused by localized cell death near the biofilm-surface boundary (see also [35]). The pattern formed by cell death is termed the cell death pattern (CDP, see Figure 2.1a, 2.1b). Localized cell death disrupts mechanical support, causing the biofilm to release compressive stress by moving into the regions cleared of cells by cell death. As the cells move into the regions opened by dying cells, they buckle and fold upwards, forming wrinkles (see Figure 1.1).

To demonstrate that wrinkling was caused by heterogenous cell death at the colony-surface interface, Asally et al. performed numerous experiments where biofilms were modified then grown under controlled conditions. The first modification was to employ a Sytox reporter, a florescent chemical that interacts with bacterial cells and fluoresces in the presence of cell death. Images of cell death in colonies revealed by Sytox staining are shown in Figure 2.1. In addition, a means for observing and calculating the movement of the biofilm material was devised. Here Asally et al. added small florescent beads to the growing biofilm and by filming the colony development from the underside and tracking the movement of these particles, spatially resolved material movement was determined. A measure called convergence (negative divergence) was computed that quantifies the inward-movement of groups of cells over a discrete grid on the plane representing the bottom of the colony. Asally et al. demonstrated that wrinkled areas and the CDP spatially overlap with areas of high convergence (see Figure 3.1) and that cell death precedes material movement.

Additionally, [3] demonstrated that ECM is required for localized cell death, thus ECM is necessary for wrinkle formation. ECM stiffness, defined by the strength of the mechanical support it provides the biofilm, is inversely related to the rate of convergence and subsequent wrinkle formation [3]. Of the two *Bacillus subtilis* strains studied in this work, WT and the mutant strain $\Delta abr\beta$, the latter shows a slower rate of convergence and wrinkle formation.

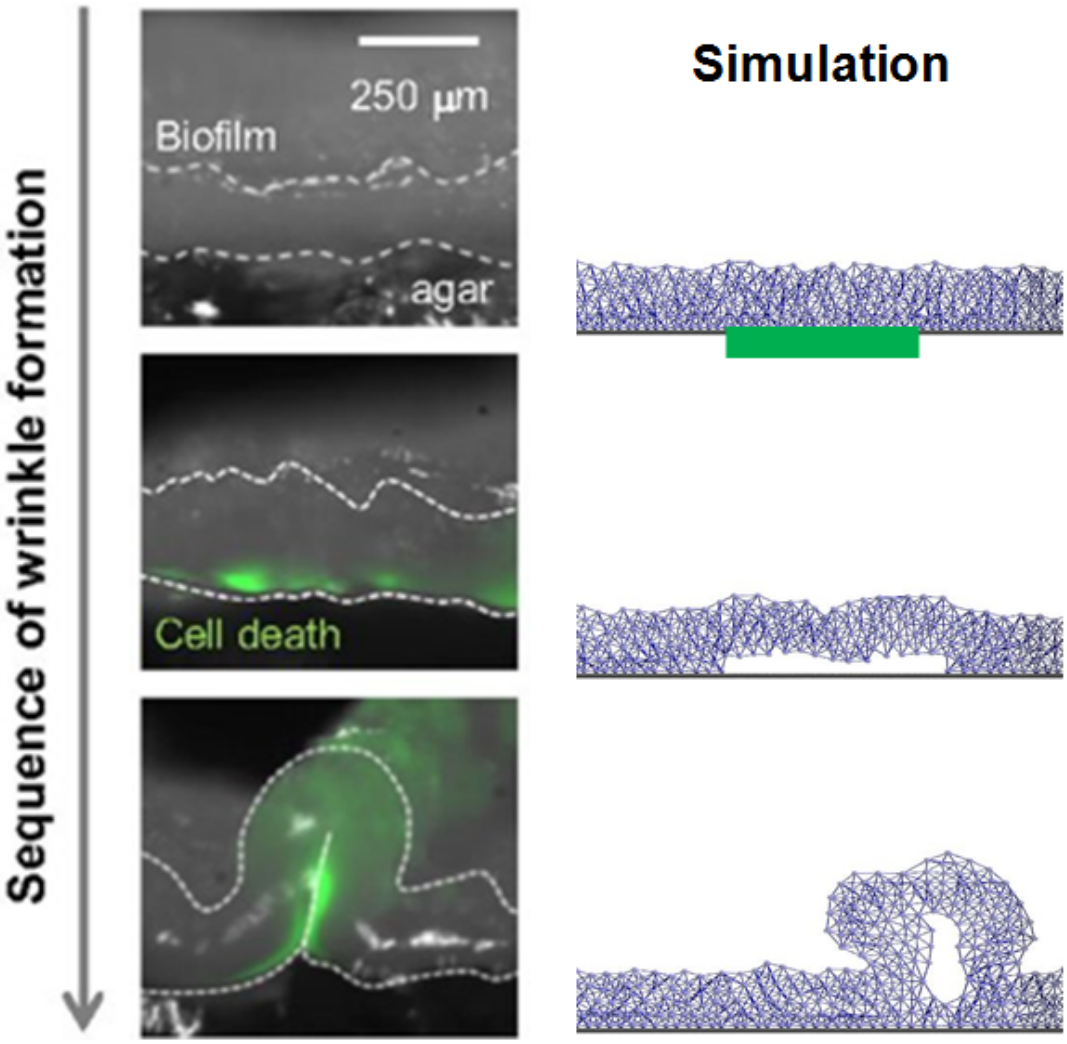


Fig. 1.1: **2D Biofilm Wrinkling** Sequence of wrinkle formation originating from cell death at the cell-substratum interface in a 2D simulation. Left column shows cross-sectional images of a wrinkle from [3] and the right column shows the simulated process. Blue particles are cells and lines between particles represent bonds. Green color shows the area of cell death.

This is because the $abr\beta$ gene product represses ECM production [40], so knocking it out leads to stiffer biofilms [3]. More force is required to wrinkle stiffer biofilms, thus the lower rate of wrinkling. Many other studies have also shown that ECM is required for wrinkle formation [37, 41, 42].

The biofilm wrinkle model described in this thesis is based on the experimental observations of [3] and a biofilm simulation system written in JAVA and extended from [8]. In this system, a biofilm is modeled by spherical particles (agents). Each particle represents a group of adjacent cells and the ECM that encases them. They each have their own individ-ual properties, e.g., mass, radius, and location. The particles exert forces on each other. Of particular interest to wrinkling is the bonding force, which exists between adjacent, linked particles and simulates the compressive and elastic forces that the ECM provides the colony. The model simulates the wrinkling that occurs in biofilms under compression after localized cell death.

Wrinkling is a multi-scale phenomenon involving at least the cellular and multicellular levels. At the cellular level, cells have properties that influence the way they interact with each other, that, through an emergent process, determines the behavior of the biofilm at the multicellular level. We chose to fit three parameters (described later) that affect the mechanical properties of particles. The behavior of the simulated colony as a whole emerges from the interaction of particles governed by the assigned values for these and other model parameters. Changing the parameters results in changes in the multicellular behavior. We fit the model by comparing its spatiotemporal convergence to the experimentally observed spatiotemporal convergence of two WT colonies and two $\Delta abr\beta$ colonies. Specifically, we partitioned the convergence of the simulations temporally into three sections (beginning, middle, and end) and compared those to three temporally consecutive intervals of real colony development.

1.2.2 Model Fitting Problem Definition

The model's behavior is controlled by a set of continuous parameters X, where $X = \{x_i, x_2, x_3...\}$. We define the parameter space as S that contains the set of all possible values

of X. Let the simulator be represented as a function *SIM* that takes as input a parameter vector X and a description of the initial conditions *Int*, and returns a sequence m_i, which are spatial configurations of the particles and their states. Let $\vec{M} = [m_1, m_2, \cdots m_T]$ be the spatiotemporal simulated data, with T being the maximum time, measured in iterations of the simulator.

$$\vec{M} \Leftarrow SIM(X, Int)\vec{M} \Leftarrow [m_1, m_2, \cdots m_T] \quad (1.1)$$

Let an experimental observation of a real biological system's configuration be e_i at time i. Then the experimentally observed spatiotemporal data is $\vec{E} = [e_1, e_2, \cdots e_T]$, with T being the maximum time, measured in hours. With the biofilm fitting considered in this work, experimental observations are sampled at 40 minute intervals and we pair each experimental interval to an iteration of the simulator, thus each iteration simulates 40 minutes of biofilm development. The data contained within each e_i will not be as detailed as the data available in each m_i due to limitations in observational and experimental techniques. In the biofilm case, the measurements taken provide spatiotemporal data in fairly close correspondence with that available through simulation. This will be expanded upon in Section 3.1

The states of both the simulation and experimental observations are represented as vectors of features. Let each multicellular feature be defined as a function $f_j(.)$ that takes either a simulated configuration m_i or a real configuration e_i and returns a real-valued array. The value may a single value, as in the case of maximal colony height, or be an array of values sampled across the spatial extent of the domain. While the function for the same feature may be implemented differently for the real or simulated data, the semantics should be the same to allow direct comparison of the values. For instance, if one of the features measures the maximum height of the biofilm colony, the simulator would extract the feature by identifying the particle with the maximal z coordinate, while with real data, the maximal height may be extracted through stereoscopic image data.

1.2.3 Multi-objective Model Fitting

The task of finding the correct values for the parameters in the model given multicellular

spatiotemporal data is cast as a multi-objective combinatorial optimization problem. Such problems proceed by searching through a parameter space $\vec{X}$ with the goal of minimizing the discrepancy (or error) between the simulated system and experimental observations. As defined above, the output of the simulation is $\vec{M}$ and the output of the experiment is $\vec{E}$ representing the spatiotemporal description of the simulation and experiment, respectively. Given a set of features f_j that take either $e_i \in \vec{E}$ or $m_i \in \vec{E}$, then, for a parameter vector X, the independent objective function o_j can be defined as the discrepancy between the feature description of the real data and the feature description of the simulated data at time t:

$$\begin{aligned} o_1(X) &= |f_1(SIM(X, Int)_t - f_1(e_t)| \\ o_2(X) &= |f_2(SIM(X, Int)_t - f_2(e_t)| \ldots \\ o_n(X) &= |f_n(SIM(X, Int)_t - f_n(e_t)| \end{aligned} \tag{1.2}$$

1.2.4 Multi-objective Optimization Algorithms

With multi-objective combinatorial optimization, the concept of Pareto optimality is useful. A solution is Pareto optimal with respect to the other solutions if no other solution exists that is better than it on all the objective measures [43] [44] [45]. Given that S represents the set of alternative parameter vectors, then formally a set of parameters $\vec{X}*$ is a Pareto optimal solution if and only if:

$$\nexists \vec{X} \in S \mid o_i(\vec{X}) \preceq o_i(\vec{X}*), \forall i, 1 \leq i \leq n \tag{1.3}$$

In other words, no solution dominates $\vec{X}*$.

The set of all Pareto optimal solutions is called the Pareto frontier and at any time during search, the Pareto frontier is the set of best solutions found so far. The Pareto frontier represents a trade-off between the quality of solutions on the different objectives.

In fitting the parameters of a model, the behavior of which is measured using multiple criteria (features), it may be difficult to tell *a priori* whether a single solution exists which dominates, i.e., is better on all the features. Thus, using multiple features as optimization

criteria is a good method of finding trade-off solutions. Hence the use of Pareto optimality. An alternative method would be to use a weighed-sum technique, in which all of the features are combined into one objective function that is the weighted sum of the various features.[45]. However, separating the criteria into different objective functions has at least three advantages: 1) The weighted-sum method requires manually defining weights specifying the relative importance of each term in the objective function. It may not be clear what the relative importance of each feature is, making defining good weights difficult. Separate objectives obviates this difficulty. 2) Solutions that perform well on a single feature but not others can be identified with separate objectives but may be missed if an aggregate objective function is used. Identifying solutions that satisfy a subset of features may elucidate the relationship between a subset of the parameters and a subset of the features. This would be impossible if a single combined function was used. 3) Similarly, in the case of a non-convex Pareto frontier a weight-sum technique is incapable of finding some points on the frontier [45].

Meta-heuristic optimization methods such as genetic algorithms and simulated annealing are superior to mathematical programming techniques for multi-objective optimization problems like the current one. Unlike many mathematical programming algorithms, meta-heuristics can handle non-convex, multi-modal, non-differentiable, and discontinuous solution spaces; are good at dealing with stochasticity; and tend to quickly converge to Pareto optimal solutions [27, 46]. Considering the current problem, the simulator is stochastic and the problem is defined multi-modally. The functions defining $\vec{o}$ are non-closed form, so they are non-differentiable (apart from estimation). Their true convexity is unknown. Furthermore, through informal experience with the simulator we have observed many regions in the parameter space resulting in no wrinkling. For models in general there may be many parameter combinations where no interesting features are present. These regions may not be helpful in fitting, thus $\vec{o}$ may be discontinuous for practical purposes.

Genetic algorithms in particular are one of the most popular approaches to multi-objective optimization [43, 47] and are well-suited for optimization problems such as the

current one. They are able to search multiple distance regions of the parameter space simultaneously, cover large portions of the space, and combine promising aspects of multiple sub-optimal solutions into better solutions via their crossover and mutation operators [48]. The algorithm described in this paper takes advantage of the ability of genetic algorithms to consider multiple solutions simultaneously by processing dozens to hundreds of simulations concurrently across a distributed cluster of CPUs.

The NSGA-II algorithm [27] was chosen for this work because it allows for parallelization, uses a fast nondominated sorting algorithm to rank solutions, uses an elitist strategy so the best solutions are never lost, and maintains population diversity without the need for a niche operator by means of crowding distance. It uses Pareto dominance to compare solutions and crowding distance to resolve ties. That is if solution X dominates solution Y, solution X is deemed the superior solution. However, if X and Y are nondominating, then it relies upon X and Y's crowding distance to resolve the tie. Briefly, crowding distance is a measure of how crowded a region in the parameter space is and is measured using the normalized Euclidean distance between solutions. If X has less solutions close to it than Y does, then X is considered the better solution.

Parents are selected in a tournament fashion utilizing the two comparators just described. A single tournament selects two parents whose genes will be combined in crossover. Multiple tournaments occur to select the entire set of parents. A tournament starts by randomly picking and comparing two solutions from the population. The better solution gets compared to a third, randomly-selected solution. This continues for the specified number of rounds, where each round is the comparison of two solutions. NSGA-II uses a single round to select the two parents.

To keep the population from exceeding a maximum size, only the best solutions are retained after each generation. Solutions are sorted using Pareto dominance ranking, which places the first-order Pareto frontier as rank 1, the second-order Pareto frontier as rank 2, and so on. The first frontier contains the nondominated solutions over the entire set while the second frontier contains the nondominated solutions over the set with the first

frontier already removed. This applies a partial ordering. In the typical implementation of the NSGA-II algorithm, solutions within a frontier are sorted using a crowding comparator, where solutions in less-crowded regions of the parameter space are preferred. This work uses a modified implementation which compares solutions based off of the product of their objectives within a frontier. If solutions are still tied, then crowding distance is applied. More information on this can be found in Section 3.1.

CHAPTER 2
MODEL AND SIMULATION

2.1 Model and Simulation

The model simulates wrinkle morphology as it progresses in time. It is implemented in a cellular simulation package written in Java, extended from Idynomics[8], a biofilm simulation program. Groups of cells and their encasing ECM are represented by spherical particles, each with its own mass, volume, and 2D or 3D location. The cells are situated in a rectangular domain ranging from a few micrometers to several millimeters on a side. The smallest unit in the simulation is a micrometer. The domain is discretized according to the resolution (r), which specifies the voxel side-length in micrometers.

Simulations can be initialized with any number of cells in a variety of shapes, e.g., a single cell, a rectangular section of a colony, a full, circular colony, etc. When initializing a simulation with a colony under compression, the particles are arranged with equal spacing, specified by the number of particles per micrometer (S_p).

2.2 Input, Output and Iterations

The simulator proceeds in discrete units of time called iterations (itr), which can represent any amount of time, but usually stand for a few minutes or hours depending upon the simulation domain. How much real time an iteration should represent is often hard to determine theoretically since it depends upon other parameters describing the model. In this work, the time calibration is performed automatically by the model validation system.

All model parameters are specified by an XML document called the protocol file. The simulator produces output at specified intervals ($outPer$), greater than or equal to itr. There are two main types of file output: agent state files and Povray [49] files. One of each type of file is output every $outPer$. The agent state files contain the properties of each

particle while the Povray files allow for visualizations of the colony and domain.

2.3 Cell Death Patterns (CDPs)

Localized cell death as observed in real biofilms is implemented in the simulation by removing particles in the bottom layers of the colony at those 2D locations were death is indicated. This is achieved by converting the experimental image showing the distribution of Sytox reporter, which fluoresces green in the presence of cell death, to a binary file where 1 signifies high Sytox. Experimental results from [3] provide CDP images for each iteration (at intervals of 40 minutes). At each time period the simulation system may load the corresponding binary file and remove the particles indicated. This is a very simplified implementation of cell death since any residue that remains from real cell death is ignored. The focus of this work is biomechanical rather than biochemical, so this simplification is reasonable. See Figure 2.1 for an example of a CDP.

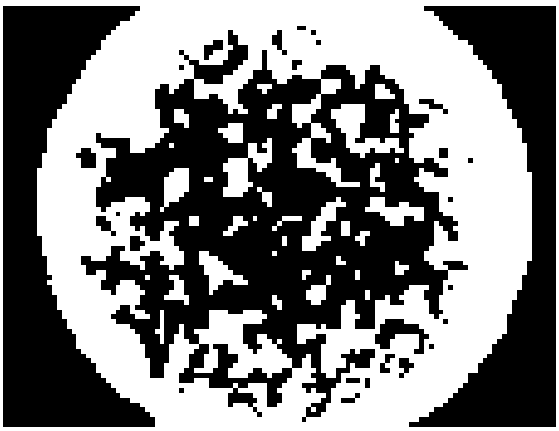
2.4 Shoving and Bonding

Particles mutually push and pull on each other according to two forces: shoving and bonding. Shoving implements the physical process of volume conservation and bonding implements the influence of EPS (extracellular polymeric substance) which "glues" the cells together.

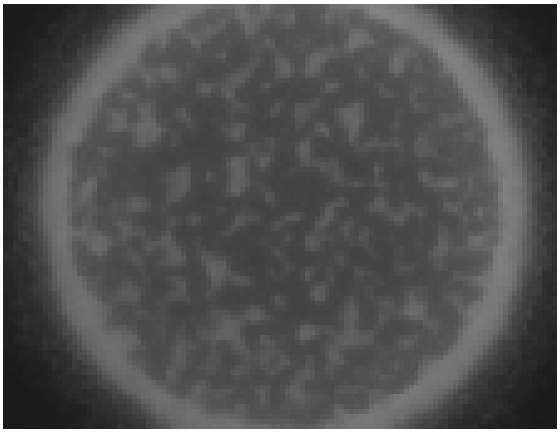
Particles compete for space and avoid overlapping by shoving. When the distance $d(\sigma_i, \sigma_j)$ between two particles σ_i and σ_j is less than $D(\sigma_i, \sigma_j)$, a repulsive force $F_{ov}(\sigma_i, \sigma_j)$ is generated to push them apart. When $d(\sigma_i, \sigma_j) >= D(\sigma_i, \sigma_j)$ there is no shoving force. The force is proportional to the overlap distance between the two particles:

$$\begin{aligned} F_{ov}(\sigma_i, \sigma_j) &= \mathrm{g}*(D(\sigma_i, \sigma_j) - d(\sigma_i, \sigma_j)), \\ D(\sigma_i, \sigma_j) &= \alpha(R_i + R_j) \end{aligned} \tag{2.1}$$

R_i is the radius of a particle σ_i. α is the shoving factor and determines the relaxed packing density of particles. $\alpha = 2.5$ for all simulations in this work. g is the shove-gain parameter and is used to control the strength of the shoving force. $g = 1$ in this work. This process is



(a) Example of binary CDP image. White indicates areas of cell death. Image was generated from Figure 2.1b using a custom contrast threshold.



(b) Grayscale sytox death reporter image of WT *Bacillus subtilis* colony, supplied by [3]

Fig. 2.1: WT ***Bacillus subtilis*** **CDP** The binary image (left) was generated from a grayscale sytox death reporter image (right) of a WT colony

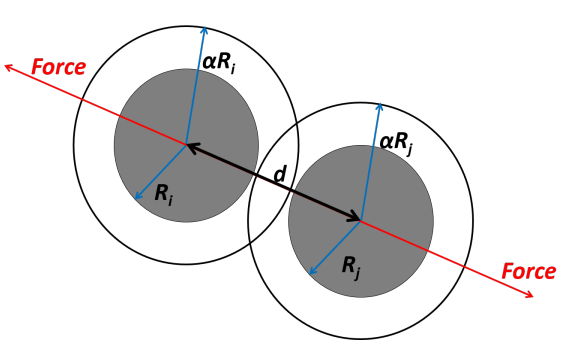
illustrated in Figure 2.2.

To model the mechanical support ECM provides the colony, nearby particles can bond with each other and with the substratum. A bond is a force between two particles or between a particle and the substratum. Bonds are the mechanism by which the biofilm maintains a compressed state and stays attached to the substratum. When ECM is turned off in the system, the particles are allowed to freely shove each other, which causes the biofilm to expand. ECM counteracts expansion.

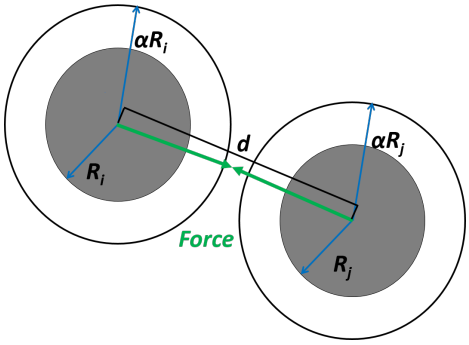
Two particles form a bond when $d(\sigma_i, \sigma_j) < a_c D(\sigma_i, \sigma_j)$. This bond is broken when $d(\sigma_i, \sigma_j) > a_d D(\sigma_i, \sigma_j)$. The bond provides a repellent force between the particles when $d(\sigma_i, \sigma_j) < D(\sigma_i, \sigma_j)$ and an attractive force when $d(\sigma_i, \sigma_j) > D(\sigma_i, \sigma_j)$. The force of the bond between two particles is given by:

$$\begin{aligned} F_b(\sigma_i, \sigma_j) &= -x_{ij} \tanh(s_b |x_{i,j}|), \\ x_{i,j} &= D(\sigma_i, \sigma_j) - d(\sigma_i, \sigma_j) \end{aligned} \tag{2.2}$$

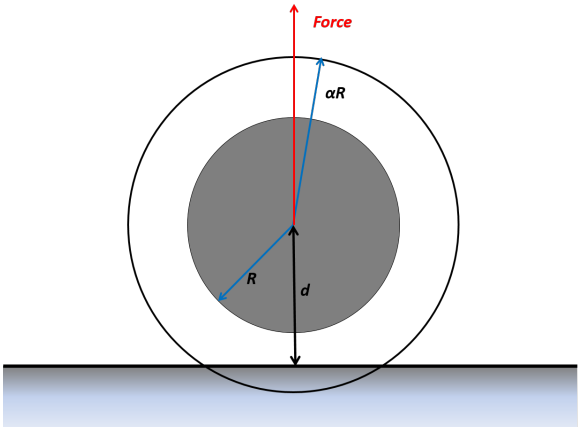
s_b is the particle-particle stiffness parameter and is used to specify the strength of the bond force between bonded particles. Increasing s_b results in stiffer ECM, which affects



(a) repellent shoving and bonding



(b) attractive bonding



(c) repellent shoving and bonding to boundary

Fig. 2.2: **Particle Shoving and Bonding:** (a) A repulsive force is applied when the distance between the particles is less than the sum of their radii times a shoving factor α. Applies to nearby particles and bonded pairs (b) An attractive force is applied between bonded particles that will pull them together if their separation distance exceeds the target distance. (c) Both forces with the same conditions also apply to the substratum boundary, but the force is only applied to the particle.

wrinkling morphology, thus it is a key parameter in the model. The bond creation factor a_c determines how close two particles need to get before they form a bond. With low values for a_c particles need to be closer together to form a bond than with higher values. The bond breakage factor a_d determines how far apart two particles can get before their bond is broken. A low value for a_d means particles don't need to drift very far apart to break. The particle-substratum bond force uses the same equation, but has its own bond stiffness, creation, and breakage factors, ss_b, as_c, and as_d, respectively. Additionally, the boolean parameter c determines when bonds can be created and broken. If c is true, then the bonds are re-evaluated each iteration itr, using the creation and breakage factors just described. Conversely, if c is false, then the bonds are only created when the simulation is initialized using a_c and as_c. In this case, new bonds are never formed and no bond is ever broken, so a_d and as_d are not utilized. In this work $c = false$ and $a_c = as_c = 1$. a_d and as_d are not used since $c = false$. s_b and ss_b are two of the three experimental parameters and their allowed values are $[0, 3]$.

Within each iteration, a force relaxation process is performed by applying multiple small adjustments to the particles' positions based upon the physical forces acting between them. That is, shoving and bonding forces are resolved for each particle S_i times each iteration. For each of the S_i rounds of the relaxation algorithm, all of the particles are iterated over in a random order. The shoving and bonding forces are calculated for a particle and then the particle and its shoving or bonded neighbor are moved in opposite directions (either towards or away from each other) a distance equal to 1/2 the force. Then the forces for the next particle are calculated and so on. Varying S_i allows for tight control over how fast a simulated biofilm wrinkles, with higher S_i resulting in faster wrinkling. By repeating the shoving algorithm each iteration, a stable particle configuration emerges, where each particle is subject to a negligible force, as the simulation progresses. This stable configuration is the final wrinkle morphology reached by the colony. See Algorithm 1 for pseudo code of the relaxation algorithm used in this research and [8] for more information on the original relaxation algorithm. S_i is the third experimental parameter and its allowed

values are [5, 20].

Algorithm 1 Force relaxation
S: Shoving iterations to perform
particles: all agents
g: shove gain, s_b: bond stiffness

```
1: procedure Shove(S_i,particles, g, s_b)
2:     c = S_i
3:     while c - - > 0 do
4:         for p_i ∈ particles do
5:             neighbors = findNeighbors(p_i)                    ▷ Nearby particles
6:             for p_j ∈ neighbors do
7:                 f = shoveForce(p_i , p_j, g)                  ▷ Uses overlap, shove gain
8:                 update movement vectors of p_i and p_j to include f/2, applied to mutually repel
9:             for p_b ∈ p_i.bonds do ▷ Bonded particles f = bondForce(p_i , p_b, s_b) ▷ Uses stiffness update
10:            movement vectors of p_i and p_b to include f/2, applied to push apart or
11:
    pull together
12:    for p ∈ particles do doMove(p) ▷ Updates position of p using movement vector, then sets
    movement vector to zero
```

A summary of these parameters can be found in Table 2.1. The parameters calibrated in this work are $X = \{s_b, ss_b, S_i\}$ and are called the experimental parameters.

Parameter Summary		
Name (Symbol)	Value	Description
shoving factor (α)	2.5	determines average particle packing density
shove gain (g)	1.0	shoving force strength
stiffness (s_b)	**[0 3.0]**	**bond force strength**
bond creation factor (a_c)	1.0	threshold for bond creation
bond breakage factor (a_d)	–	threshold for removing bonds
boundary stiffness (ss_b)	**[0 3.0]**	**boundary bond force strength**
boundary bond create factor (as_c)	1.0	threshold for boundary bond creation
boundary bond breakage factor (as_d)	–	threshold for removing boundary bonds
(c)	false	should bonds be recalculated each iteration?
shove cycles (S_i)	**[5 20]**	**number of shoving cycles per iteration**

Table 2.1: **Biofilm Model Parameters:** Bold indicates the experimental parameters. Their allowed ranges are given. For the other inputs, the exact values used are shown. "–" indicates unused parameters.

CHAPTER 3
METHODS AND EXPERIMENTAL SETUP

3.1 Search Method

All parameters, including the experimental values we are attempting to calibrate, are set through the XML protocol file. The search starts by reading a template protocol file, which contains placeholders for the experimental parameters and exact values for the others. Each placeholder specifies the type of variable, (i.e., real number, integer, or boolean), and the minimum and maximum value the parameter can take. Together, the placeholders define the parameter space S. The search algorithm uses the template to create a protocol file for each simulation to be run, inserting the separate parameter values that it chose for each. That is, some $X \in S$ is defined for each simulation to be performed and inserted into the protocol file for that run. Simulation output, e.g., agent state and Povray, files are stored in the same file location as the simulation's protocol file.

The algorithm is distributed across multiple machines and cores. The simulation program itself is single-threaded, but the search algorithm runs multiple instances in parallel. There are three types of processes: one master, one overseer, and many workers. They share data via a local network and communicate through a shared file database called the roster. The roster stores folder locations of simulations (UNC paths) and their states: *pending*, *inprogress*, and $finished$. The master runs the genetic search algorithm, the worker processes run simulations, and the overseer process periodically checks for failed simulations. Simulations sometimes do not complete due to workers failing. The overseer marks failed simulations so the master knows not to include them. Each worker process runs simulations in serial and writes output for each to the file location for that simulation as specified by the roster. Workers look for pending simulation records from the roster. When a worker finds one, it updates the record as *inprogress*. When it is finished with the simulation, it

updates the record to $finished$. The processes utilize a simple read/write locking system to avoid conflicts.

The algorithm starts by reading in the spatiotemporal experimental data $\vec{E}$; extracting the specified features, e.g., convergence, by applying a feature function f to $\vec{E}$ for each specified feature; and reading the protocol template file from which it pulls the experimental parameter placeholders S and all other inputs. It stores all of this information for later use. Here, we define f generically for generality, but it must be explicitly implemented for each desired feature. For each f programmed to extract some feature from $\vec{E}$, a separate function with the same semantics must be defined to extract the same feature from the simulation's spatiotemporal data $\vec{M}$. For example, we have programmed two versions of f to extract convergence, one for the experimental data and a separate one for the simulated data.

The initial population of the specified size is generated by randomly selecting parameter values from within their respective ranges, i.e., random Xs are created by selecting values from S. These are added to the population. Each subsequent generation is created from the previous through selecting parents, applying crossover, and then mutating them. Parent selection is done using Pareto dominance and crowding distance-based tournament selection, as briefly described in section 1.2. In this work, single-point crossover and bitwise mutation are used. All parameter types are binary-encoded. Real-valued variables are encoded to the specified number of significant decimals places. In this work, s_b and ss_b are encoded to 4 decimal places and S_i is an integer.

To keep the population from growing indefinitely as more children are produced, it is truncated to some maximum, which is typically the size of the initial population. In NSGA-II's standard implementation, solutions are ranked by Pareto dominance and then ranked by crowding distance within each Pareto tier, as described in section 1.2. In our modified implementation, we rank the solutions within each tier by the product of their objectives, placing smaller products first. Combining Pareto ranking and product ranking provides a total ordering which prefers nondominating solutions in the first place but also favors solutions with the overall lowest objectives. The new population is then truncated to

only include the best solutions. Depending upon the solutions and the maximum population amount, this may be a subset of the first Pareto frontier or include multiple Pareto frontiers. The final set of solutions returned by the algorithm are the first Pareto tier. The complete history of solutions evaluated and the members in the population at each generation are saved to a file to facilitate future analysis.

At each generation the master process creates and copies the necessary files and folders for each offspring simulation, including an encapsulating folder which will contain all simulation files, the CDP binary image file, a protocol file generated from the template with the chosen parameter values inserted, and the sub-folders where simulation output files will be stored. The master appends the UNC path of the encapsulating folder to the roster file along with an *inprogress* state. This lets any available worker know that it is free to run this simulation.

Next, the master continually looks for finished simulations by reading the roster for those with a state of $finished$. It reads the simulation output files and extracts the specified features from this data, e.g., convergence, by applying f to $\vec{M}$ for each specified feature. After extraction the features are output to plain text files so they can be conveniently visualized and analyzed later. Finally, the master generates errors and continues to the next generation, in a partially-asynchronous way. Although the master adds a certain number of simulation offspring to the roster each generation, it does not wait for all of these to be finished; instead, it waits for a threshold number to complete before moving on. It keeps track of all previously-unfinished simulations it has written to the roster in the current and previous generations and continues when $CONT_{THRESH}\%$ of these have finished and had their features extracted. Given, $lastFin_{count}$, the total number of simulations completed this generation and $unfin_{count}$, the number of simulations not finished in previous generations plus those added this generation, the algorithm evaluates these simulations and moves on to the next generation when $100(lastFin_{Count}/unfin_{count}) > CONT_{THRESH}$. Note, the counts for $lastFin_{count}$ and $unfin_{count}$ do not contain simulations finished in previous generations. The algorithm terminates once one or more of its exit conditions have been

met. These currently include, the maximum number of solution evaluations or generations exceeded and a time limit. It checks ahead to see if the next generation will be the last. If so, it waits until all *pending* and *inprogress* simulations are finished before doing the final evaluation and terminating.

Finally, the master creates $lastFin_{count}$ offspring in the next generation, just enough to replace those finished and extracted in the last generation. This helps ensure that all machine resources are fully utilized. Setting $CONT_{THRESH}$ to a value less than 100 in-creases algorithm speed because it does not have to wait for the slowest-running simulations to finish. A simulation runs slower than others depending upon its parameters, machines resources, and network latency. Ideally, $CONT_{THRESH}$ is set to the lowest value possible while still allowing for each generation to contain some minimum number of simulation solutions. However, if $CONT_{THRESH}$ is too low, the genetic algorithm tends to favor faster running simulations. We set $CONT_{THRESH} = 90$. At this value the algorithm does not seem to favor faster simulations, but it does not have to wait for the few slowest-running ones. We did not rigorously test this value and it will likely be the subject of future testing.

Before continuing to the next generation the algorithm evaluates the newly-finished simulations by comparing their features to the experimental features, generating an error for each and turning these errors into objectives (see sections 1.2.2 and 1.2.3 and equation 1.2). Each feature either applies to the whole simulation or some subset of its iterations, e.g., convergence between iteration 3 and 4, as defined by the function f for that feature. These errors are either directly used as objectives (as in equation 1.2) or turned into objec-tives by combining and/or averaging them in some way, e.g., averaging the errors over all iterations into a single objective. Specifying new feature functions and different calculations for generating errors and objectives is achievable by overriding and adding classes to the JAVA code.

See algorithms 1 - 6 and global variables 1 for pseudo code of what occurs in a single generation of search, other than the first. In part, these algorithms contain pseudo code of the modified version of NSGA-II. The code implementing and applying the other available

genetic algorithms is slightly different and not included in this text.

Global Variables 1 Variables for algorithms 1 - 6

STATE{pending, in progress, finished}
CONSTANTS {
$CONT_{THRESH} = 90$: Determines when to continue to the next generation
$WAIT_{TIME}$: time to wait between checks for newly finished simulations
}
ARGUMENTS {
$roster_{file}$: UNC path to the roster file
$templateProt_{file}$: Template protocol file used to create a specific protocol file for each simulation
out_{fol}: UNC path where all simulations will be stored
max_{pop}: The maximum number of solutions in the population
}
OTHER VARIABLES {
$features_{exp} = f(\vec{E})$: The extracted experimental features.
$lastFin_{count}$: The total number of solutions finished and extracted at the last generation
$unfin_{count}$: The total number of unfinished solutions from the current and previous generations }

Algorithm 1 Processes a single *generation* >= 2
pop: current population, the best so far.
isLast: *true* if this is the last generation.

```
1: procedure iterate(pop, isLast)
2:     offspring = createOffspring(pop)
3:     sol_fin = evaluateAll(offspring, isLast)
4:     pop.addAll(sol_fin)
5:     truncate(pop, max_pop)                ▷ Ranks sols, incls. max_pop best
```

Algorithm 2 Selects parents from population. Applies crossover and mutation to spawn offspring

```
1: procedure createOffspring(pop)
2:     offspring = emptylist
3:     while offspring.size < lastFin_count do
4:         parents = tournSel(pop)                ▷ Pareto tournament▷
5:         children = evolve(parents)             ▷ crossover,mutation
6:         offspring.addAll(children)
       return offspring
```

3.1.1 Experimental Data

The experimental data we used to fit the model was supplied by Dr. Grol M. Sel at

Algorithm 3 Evaluates solutions as they are finished.

procedure evaluateAll($solutions$, $isLast$)
    $sol_{fin-All} = emptylist$
    sendNewSimulations($solutions$)
    $unfin_{count}$ += $solutions$.size
    $tmpUnfin_{count} = unfin_{count}$
    while 100($lastFin_{count}$ / $unfin_{count}$) < $CONT_{THRESH}$ or ($isLast$ and $tmpUnfin_{count} >$ 0) **do**
        sol_{fin} = getSolsAsFin()
        evaluate(sol_{fin})
        $tmpUnfin_{count}$ -= sol_{fin}.size
        $sol_{fin-All}$.addAll(sol_{fin})
        $lastFin_{count} = sol_{fin-All}$.size
    $unfin_{count}$ -= $lastFin_{count}$
    return $sol_{fin-All}$

Algorithm 4 Creates folders for output, writes protocol, and add solutions to the roster.

procedure sendNewSimulations($solutions$)
    $simOut_{fols}$ = createSimFiles($solutions$, $templateProt_{file}$, out_{fol}) ▷ Creates sim folders, protocol
    appendRoster($roster_{file}$, $simOut_{fols}$, $STATE.pending$) ▷ Updates roster

Algorithm 5 Gets 1 or more finished solutions from roster

procedure getSolsAsFin
    $sol_{fin} = emptylist$
    while sol_{fin}.size == 0 **do**
        sol_{fin}= readRoster($roster_{file}$, $STATE.finished$) ▷ Gets finished solutions from roster
        sleep for $WAIT_{TIME}$
    return sol_{fin}

Algorithm 6 Gets data, extracts, evaluates, and generates objectives for finished solutions

procedure evaluate(sol_{fin})
    for $sim \in sol_{fin}$ **do**
        $\vec{M}$ = readData(sim) ▷ Reads sim output
        $features_{sim}$ = f($\vec{M}$) ▷ Extracts features
        $errors$ = calcErs($features_{exp}$, $features_{sim}$) ▷ Compares features
        setObjs(sim, $errors$) ▷ Objectives from errors

the University of California in San Diego, La Jolla, CA. In this work, tracking beads and Sytox death reporters were placed on the underside of colonies and images were taken every 40 minutes. These images are 672 x 512 px with $142px \approx 1mm$. The images of the beads on the colony bottom allowed them track the 2D positions of groups of cells in adjacent frames. Using these trajectories, we interpolated a velocity field over a regular mesh for

every pair of frames. Spatiotemporal convergence was calculated as the flux of these velocity fields. Thus, we generated a mesh of scalar values for each pair of adjacent frames and this is the convergence. A positive value for a tile in this mesh indicates a net movement of particles into the tile. Negative values were ignored. To generate the CDPs, we converted the grayscale images of the Sytox death reporters to binary images. See Figure 2.1 for an example CDP.

3.1.2 Parameter Fitting

In the following, t_s indicates iteration(s) of the simulations and t_e indicates frame(s) of the experimental images. Simple colonies were simulated using approximately $N = 750,000$ particles arranged in a cylinder with initial radius $R = 2078\mu M$ and height $H = 100\mu M$. The colonies were placed in a domain of size $nJ * nK * nI = 394 * 300 * 33$ voxels with voxel side length $r = 12\mu M$, making for a simulated domain size of $Y * Z * X \approx 4725 * 3600 * 396\mu M$. The height, X, was set sufficiently high to guarantee the domain could accommodate the tallest wrinkles while the side lengths Y and Z were set to match the dimensions of the supplied images. R was set to the approximate radius of the colonies observed in the images at $t = 20$, the time slice used for the CDP. The particles were equally spaced with $S_p = 0.1 particles/\mu M$ at $t_s = 0$. The particles were allowed to shove each other for two simulation iterations, to allow for their arrangement to become disorganized, before applying cell death at $t_s = 2$. Figure 3.1 illustrates the basic experimental process of applying the CDP and convergence analysis of the resulting wrinkle for a single iteration.

We ran the simulations until $t_s = 18$ and compared simulation convergence at times $t_s = [3, 18]$, inclusive to experimental convergence at image frames $t_e = [29, 44]$. Specifically, we compared the convergence of adjacent pairs of simulation iterations to adjacent pairs of experimental image frames, i.e., $t_s = [3, 4]$ to $t_e = [29, 30]$, $t_s = [4, 5]$ to $t_e = [30, 31]$,... $t_s = [17, 18]$ to $t_e = [43, 44]$. The experimental images were taken 40 minutes apart, so the portion of the simulations compared to experimental data ($t_s = [3, 18]$) represent 10 hours of dynamic colony morphology. The lag time of 9 frames (representing 6 hours) between the frame used for the CDP and the first experimental convergence frame was informed

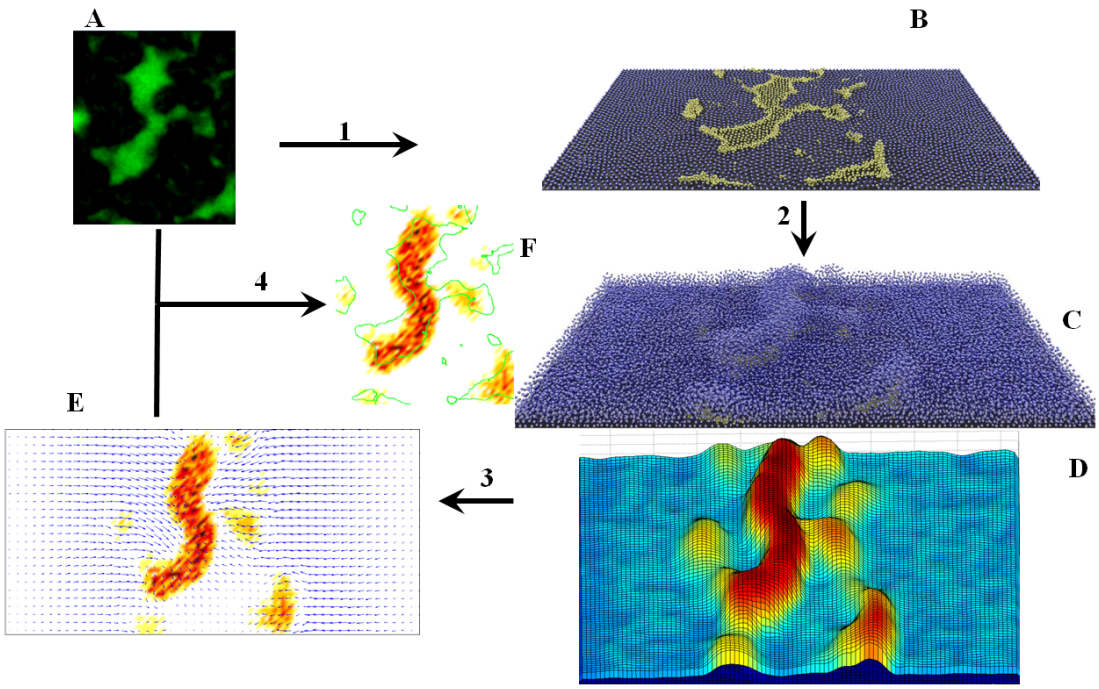


Fig. 3.1: **CDP, Convergence and Wrinkling** Schematic of connection between CDP application, convergence, and wrinkling : A) Cell death pattern (CDP) from part of a colony adopted from [3]. B) CDP mapped to the bottom layer of a colony in which the cells are in a quasi stable state. Note, only the bottom layer is show. C) Buckling over the CDP area gives rise to the wrinkles. D) Reconstructed surface of biofilm. E) Velocity vectors and convergence of vector fields computed from material movement. F) Spatial correlation of CDP and wrinkles. Colors indicate magnitude, with red indicating the largest magnitude.

by the observations of [3]. They observed that the maximum overlap between a CDP and convergence occurred with a lag of 6 hours.

Although the simulations used a resolution of $12\mu M$, a coarser resolution of $\approx 42\mu M$ was used in calculating convergence. This reduced the convergence mesh to $113 * 86 tiles$. We used a matching mesh size in calculating experimental convergence from the bead trajectories. We decided upon this grid size from informal study of experimental convergence: this size was coarse enough to reduce noise while large enough that the overall convergence pattern was not lost. Moreover, since the tracking beads were only placed on the bottom of the colonies (due to practical limitations), we only have convergence data for the cells near the colony-substratum interface. Thus, we only compared simulated convergence of the particles in the bottom layer to experimental convergence. Hence, the 2D convergence mesh. For the simulation, the bottom layer was defined as at or below $24\mu M$, $2r$ or approximately $1/4$ the initial colony height.

Model fitting was performed over four data sets, each from a different colony. Two of these were wild type (WT) colonies and two were $\Delta abr\beta$ colonies. Four repeats of the genetic algorithm were executed for each colony. For each run, the initial population was $P = 50$ and it was allocated 60 hours of wall clock time, which resulted in approximately 30-40 generations. The number of generations processed in this time varied due to differences in simulation runtime, which depends on the exact experimental parameter values.

3.1.2.1 Algorithm Validation with Synthetic Data

Before applying the genetic algorithm to fit the model against experimental data, it was validated against the output of a simulation with specific parameters values known to induce wrinkling. We call this simulation the target. Aside from experimental parameters, the initial conditions of the target and test simulations were identical. Since the parameters of the target simulation were exactly known, we were able to test the ability of the genetic algorithm to find minimum error solutions, i.e., those with nearly identical parameters. We validated against four different target simulations and ran the algorithm ten times on each, for a total of forty validation runs. To decrease the processing time of validation, we

simulated small square sections of colonies of size $100 * 100 * 33 voxels$ $(1200 * 1200x396\mu M)$ using approximately 80,000 particles, instead of full colonies. We used CDP sections of the same size. To ensure interesting wrinkle patterns, the CDPs for validation were generated from subsections of the experimental Sytox death reporter images, two from the $\Delta abr\beta$ colonies and two from the WT colonies. This is opposed to using completely synthetic CDPs, which would be simpler and likely generate less interesting morphologies. Besides these differences in simulation domain size, all simulation and genetic algorithm parameters were identical to those used in model fitting.

3.1.3 Fitness Functions, Errors, and Objectives

We compared the simulated convergence of consecutive iteration pairs to the convergence of consecutive experimental image frame pairs. The meshes were compared tile by tile. We define a feature function for convergence f_c that takes either $(e_t, e_{t+1}) \in \vec{E}$ or $(m_t, m_{t+1}) \in \vec{M}$ and calculates the simulated or experimental convergence mesh, respectively, between time slices t and $t+1$. Then, the error between convergence at simulated time t_s to $t_s + 1$ and convergence at experimental time t_e to $t_e + 1$ is calculated by:

$$Er_C(t_s, t_e) = \frac{\sum_{j=0}^{cJ} \sum_{k=0}^{cK} |f_c(m_{t_s}, m_{t_s+1})[j,k] - f_c(e_{t_e}, e_{t_e+1})[j,k]|}{(cJ * cK)} \tag{3.1}$$

where cJ and cK are the dimensions of the convergence meshes and $f_c(...)[j,k]$ retrieves the value of the mesh at tile location (j,k).

We then partitioned the errors into temporal sections, each of size T_C. The average of the errors over each section became the objectives supplied to the genetic algorithm. A single objective over a section of the iterations t_s to $t_s + T_C$ and frames t_e to $t_e + T_C$ is calculated by:

$$\sum_{i=t_s, j=t_s}^{t_s+T_C-1, t_e+T_C-1} \frac{Er_C(i,j)}{T_C} \tag{3.2}$$

Note that the number of simulation iterations and image frames must be the same.

We set $T_C = 5$, splitting the fifteen iterations into three sections, the beginning ($t_s = [3, 8]$), middle ($t_s = [8, 13]$), and end ($t_s = [13, 18]$) of the simulation.

CHAPTER 4

RESULTS, DISCUSSION, AND CONCLUSION

4.1 Results

We first show the results of validating the algorithm on synthetic data and then proceed to the outcome of parameter fitting for experimental data.

4.1.1 Validation on Synthetic Data

To demonstrate that the search algorithm is capable of finding minimum-error solutions and correct model parameter values, we applied it to the output data of four simulations run with known parameter values (synthetic data). We call these simulations the targets. Table 4.1 presents a summary of the parameter values found in these runs. As the algorithm proceeds, the population gets closer to the true Pareto-frontier in its objectives and closer to the target parameter vector. See Figure 4.1 for a visualization of the approximate Pareto frontier as the population approaches minimum error and Figure 4.2 for an illustration of the population progressing closer to the target in the parameter space. These improvements can be seen directly by comparing color-maps of target convergence to the convergence of

			CDP 1		**CDP 2**	
		Targets	Found	% Error	Found	% Error
Target 1	s_b	0.03	$\mathbf{0.0289} \pm 0.0010$	3.6	$\mathbf{0.0320} \pm 0.0019$	6.7
	ss_b	1.0	$\mathbf{1.4480} \pm 0.2096$	44.8	$\mathbf{1.9739} \pm 0.3732$	97.4
	S_i	16	$\mathbf{13.1304} \pm 0.3035$	17.9	$\mathbf{14.4545} \pm 0.2073$	9.6
Target 2	s_b	0.1	$\mathbf{0.0852} \pm 0.0040$	14.8	$\mathbf{0.0951} \pm 0.0017$	4.9
	ss_b	2.0	$\mathbf{1.1942} \pm 0.2560$	40.3	$\mathbf{1.0088} \pm 0.1381,$	49.6
	S_i	9	$\mathbf{8.1667} \pm 0.1667$	9.3	$\mathbf{8.6667} \pm 0.1290$	3.7

Table 4.1: **Synthetic Data: Discovered Parameter Values:** Summary of the results for the four validation runs using synthetic data. Each run was repeated ten times. The parameter values are averages across all Pareto-optimal solutions on all repeats. The errors are calculated as the average absolute difference between the best solution parameter values and the target values. Note the accurate and consistent solutions for parameters s_b and S_i.

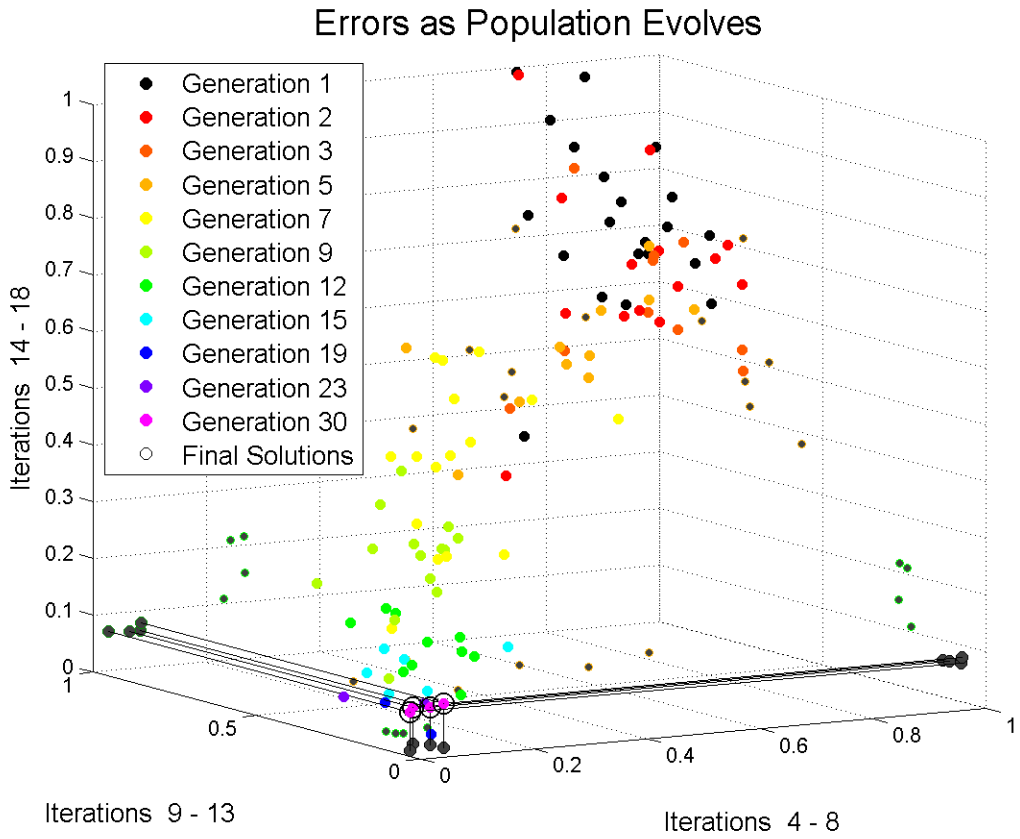


Fig. 4.1: **Synthetic Data: Objective Space** Scatter plot that visualizes the 3-dimensional Pareto frontier of objectives as it improves over the generations. Each dot is a solution. The three axis correspond to the three objectives, normalized from 0 to 1. The generations shown were chosen to give a good representation of the algorithm's progression towards the final (Pareto-optimal) solutions. 2D projections (gray with colored edges) are shown for the 5th, 12th, and final-generation solutions. Lines connect the final solution markers and their 2D projections as well. Many points belong to more than one generation but only the latest generation is shown. Data is from a single validation run.

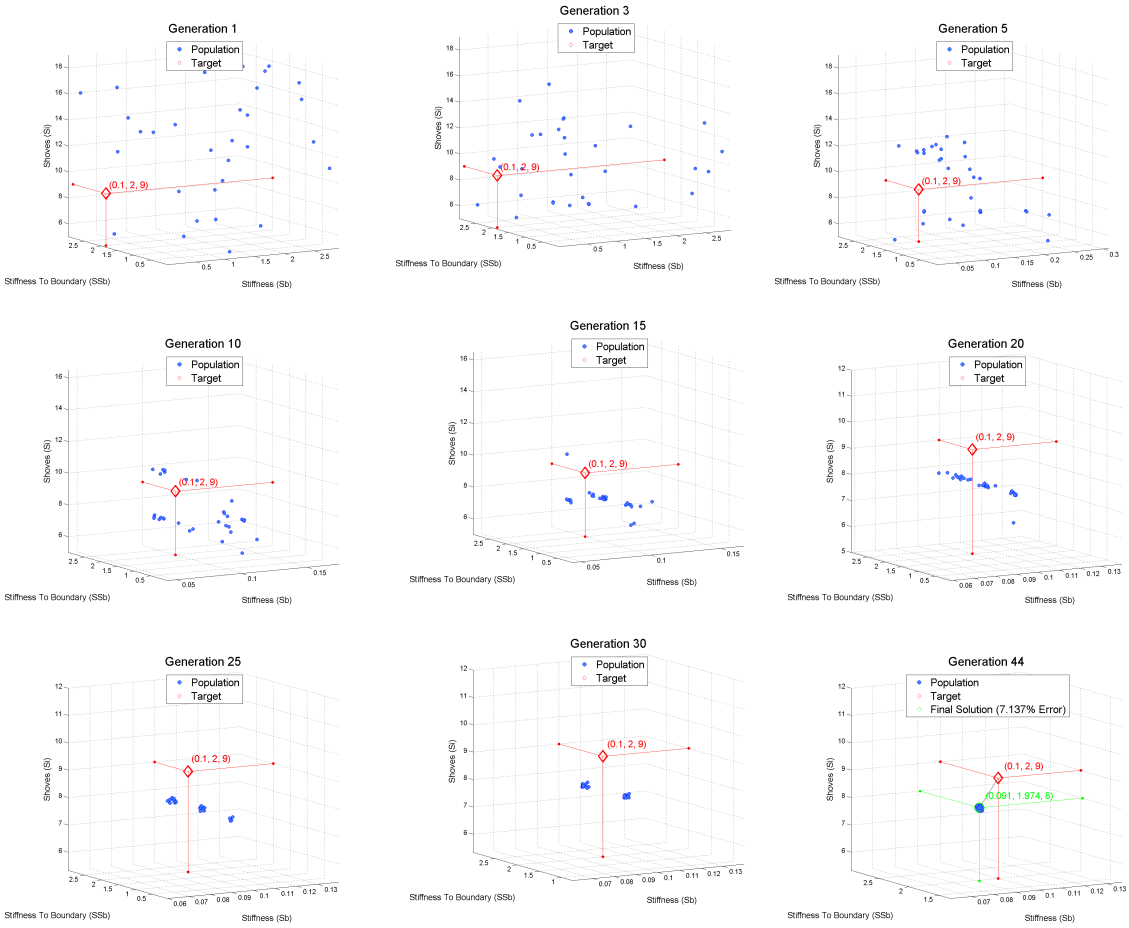
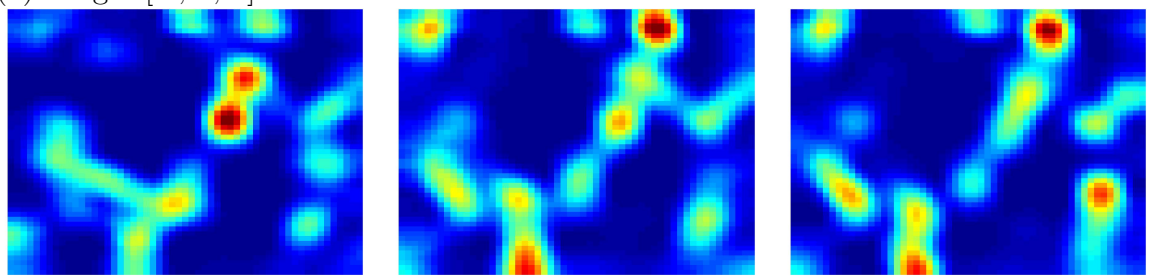


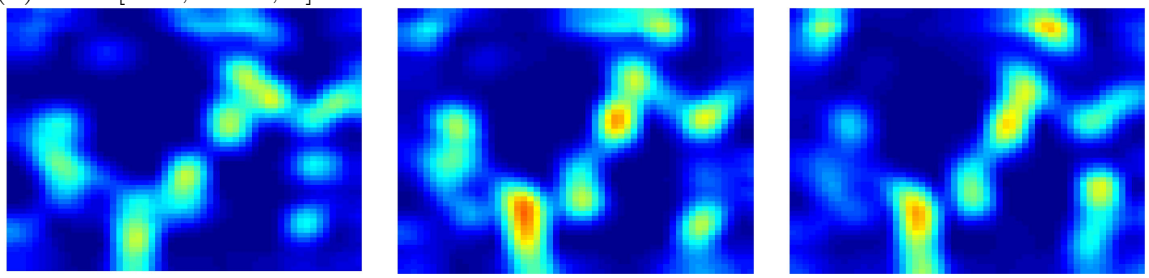
Fig. 4.2: **Synthetic Data: Parameter Space:** The Parameter space as the algorithm approaches solutions near the synthetic target. The target parameter values are shown in red. Lines connect the 3D locations of the target and the final solution to their respective 2D projections to better show their locations. The final solution shown was the best solution found on this run of the algorithm. It is a Pareto-optimal solution with the minimum product of objectives. Data is from a single validation run. Points with very similar values were perturbed slightly to decrease overlap. Note the scales of the axes change in the plots to focus-in on the interesting region.

(a) Target [.1, 2, 9]



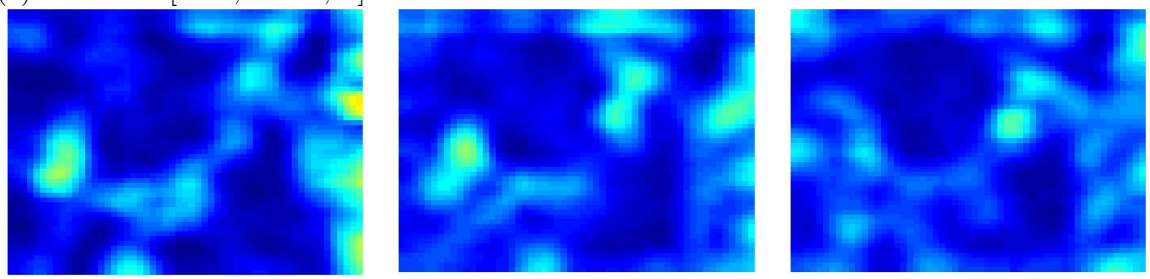
(a) Iterations 3-8 (b) 8-13 (c) 13-18

(b) Best [.091, 1.974, 8]



(d) Iterations 3-8 (e) 8-13 (f) 13-18

(c) Random [.723, 2.882, 5]



(g) Iterations 3-8 (h) 8-13 (i) 13-18

Fig. 4.3: **Synthetic Data: Convergence Feature:** The convergence feature of the target and example solutions from a validation run of the search algorithm. Each image shows the combined convergence for the simulation beginning (iteration 3 - 8), middle (iteration 9 - 12), and end (iterations 13 - 18), respectively. (a) The target convergence values generated with the parameter vector, [.1, 2, 9]. (b) The convergence values for the best solution found by the algorithm, parameter vector [.091, 1.974, 8]. Note the close correspondence between the target and the best solution found. (c) Convergence for the three time periods from an initial random solution with parameter vector [.723, 2.882, 5]. Note the lack of correspondence between these convergence values and the target. The best solution is Pareto optimal with the lowest product of the three objectives. All images use the same color scale with red being high convergence and blue being low convergence.

a poor solution and a good solution as seen in Figure 4.3. The improvements can also be visualized in Figure 4.4, which shows the surfaces of the target biofilm, a poor solution, and a good solution. Figure 4.5 provides a summary of the performance for the best performing solution at each generation. The error as a function of generation is given in the objective space in Figure 4.5(a) and the parameter space in Figure 4.5(b). These figures show that the overall best solution at each generation improves over the generations.

4.1.2 Full Colony Experimental Parameter Fitting

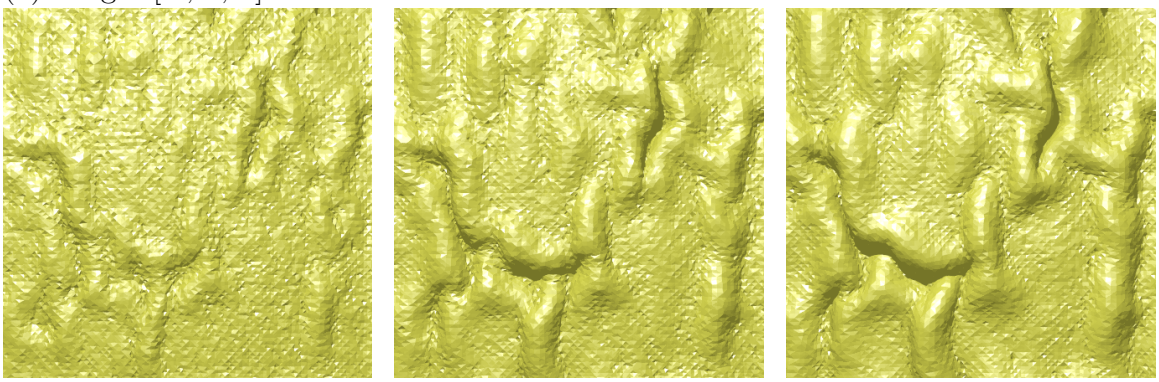
The developed search algorithm was applied to two $\Delta abr\beta$ colonies and two WT colonies. For simplicity, we refer to these four colonies as $\Delta abr\beta_1$, $\Delta abr\beta_2$, WT_1, and WT_2. Parameter searches for each colony were repeated four times for a total of sixteen runs. The runs were repeated because the search algorithm is stochastic, with each run returning slightly different parameter vectors. The best and average values found for the three experimental parameters, s_b, ss_b, and S_i are summarized in Table 4.2. The best solu-tion on a given run or among many runs is defined as the solution with the lowest product of errors among the final Pareto-optimal solutions.

As the algorithm progresses, the population evolves to include better solutions, as seen in Figure 4.6 which shows the errors decreasing during the fitting to colony $\Delta abr\beta_1$. Improvements in particle convergence for the same colony is shown in Figure 4.7, which compares the convergence of the real $\Delta abr\beta_1$ colony to the simulated convergence of a poor solution and the overall best solution.

4.2 Discussion and Conclusions

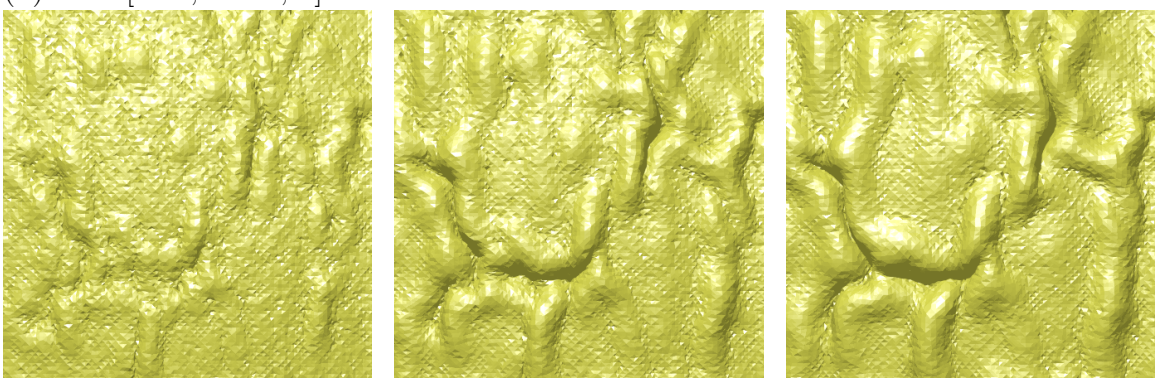
Multiscale models are widely used to gain understanding of and make predictions about dynamic biological systems spanning multiple spatiotemporal scales. These models typically rely upon input parameters to control their behavior. In particular, multicellular models rely upon parameters that control cellular and sub-cellular physiology, which in turn impacts multicellular behavior. Finding parameter values such that the resulting multicellular outcomes of the simulated system reproduce experimentally observed outcomes is crucial

(a) Target [.1, 2, 9]



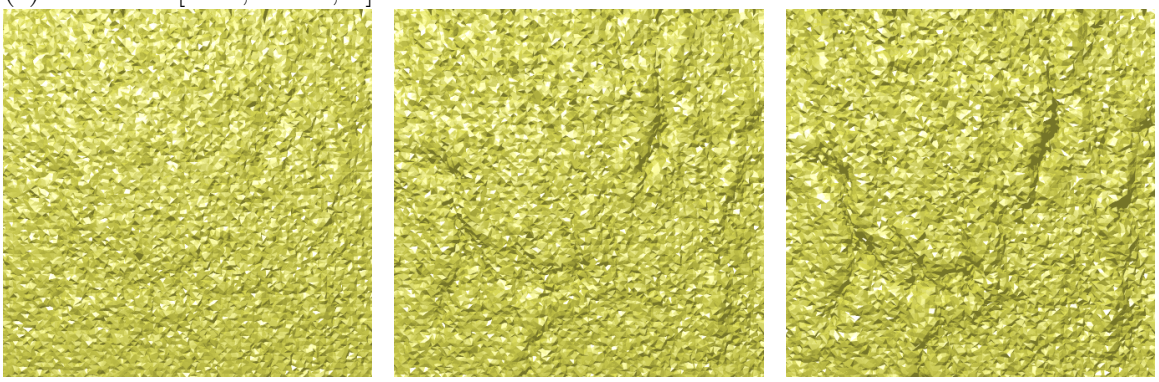
(a) Iteration 6 (b) Iteration 12 (c) Iteration 18

(b) Best [.091, 1.974, 8]



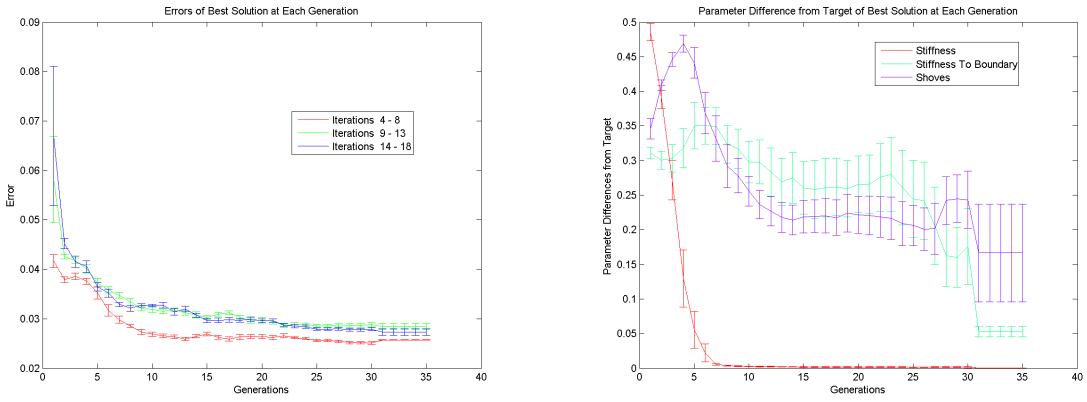
(d) Iteration 6 (e) Iteration 12 (f) Iteration 18

(c) Random [.723, 2.882, 5]



(g) Iteration 6 (h) Iteration 12 (i) Iteration 18

Fig. 4.4: **Synthetic Data: Surface Visualization:** Visualization of the surface of the target and example solutions from a validation run of the search algorithm. Each image shows the surface for the simulation at iteration 6, iteration 12, and iterations 18. (a) The surface generated with the target parameter vector, [.1, 2, 9]. (b) The surface for the best solution found by the algorithm with parameter vector [.091, 1.974, 8]. Note the close correspondence between the target and the best solution even though only convergence error was minimized. (c) Surface from an initial random solution with parameter vector [.723, 2.882, 5]. The best solution is Pareto optimal with the lowest product of the three objectives.



(a) Objective Space Error

(b) Parameter Space Error

Fig. 4.5: **Synthetic Data: Improvement of Best Solution:** (a) Error between the best solution and the target solution for the three objectives, beginning (iterations 4 - 8), middle (iterations 9 - 13) and end (iterations 14 - 18) as a function of generation. (b) Absolute difference between the correct parameter values (synthetic target) and the best solutions' parameter values as a function of generation. The best solution is the one that is Pareto optimal and with the lowest product of the three objective errors. Error bars indicate the standard deviation over 10 repeated algorithm runs.

	WT			***Δabrβ***		
	WT_1	WT_2	Combined	$\Delta abr\beta_1$	$\Delta abr\beta_2$	Combined
s_b	0.0235 ± 0.0007(**0.024**)	0.0201 ± 0.0017(**0.020**)	0.0213 ± 0.0012(**0.020**)	0.0522 ± 0.0029(**0.057**)	0.0393 ± 0.0059(**0.034**)	0.0479 ± 0.0032(**0.057**)
ss_b	0.0475 ± 0.0078(**0.054**)	0.0586 ± 0.0072(**0.071**)	0.0546 ± 0.0054(**0.071**)	0.1625 ± 0.0098(**0.155**)	0.1122 ± 0.0144(**0.092**)	0.1457 ± 0.0105(**0.155**)
S_i	17.6250 ± 0.3750(**17**)	18.9286 ± 0.1269(**19**)	18.4545 ± 0.2052(**19**)	11.6250 ± 0.2631(**12**)	12.0000 ± 0.7071(**14**)	11.7500 ± 0.2787(**12**)

Table 4.2: **Experimental Data: Discovered Parameter Values:** Summary of the experimental parameter values found by the search algorithm. Shown for each of the four colonies (two of each strain) are the collated and averaged values of the Pareto-optimal solutions over the repeated runs (four). The combined column lists the combined averages over the two colonies from that strain, i.e., among eight runs. Shown in bold is the best solution among the respective group (lowest product of objectives).

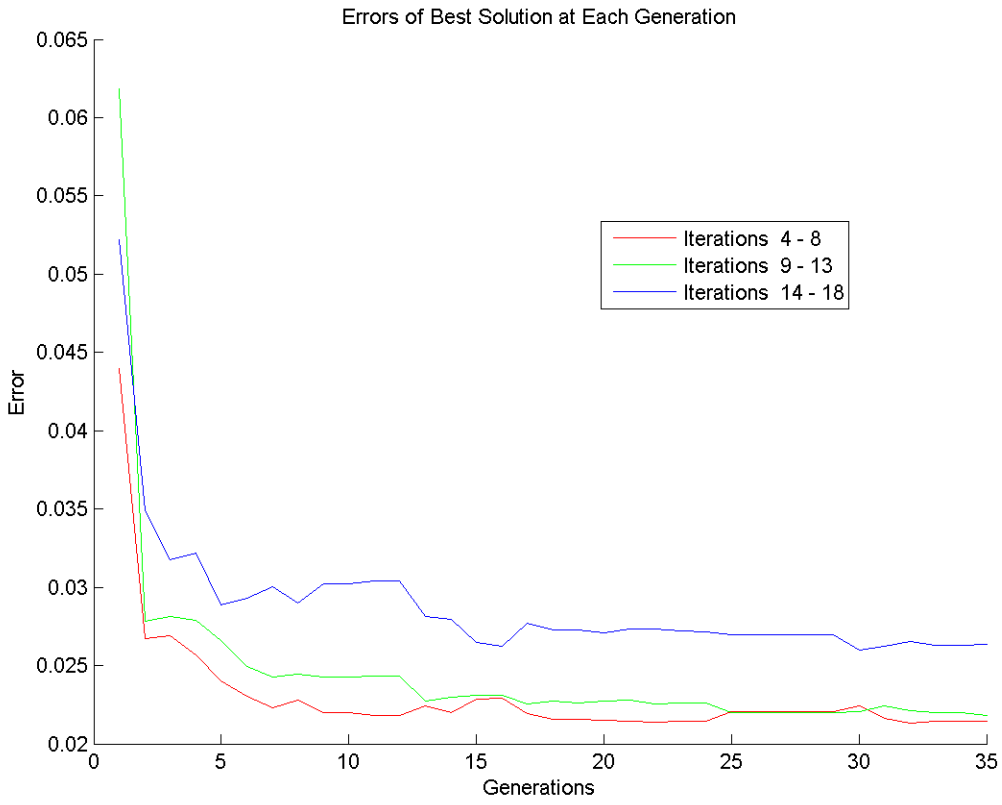
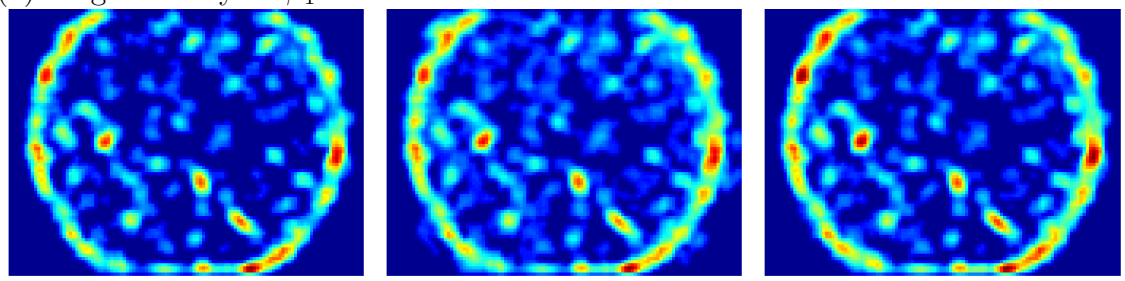


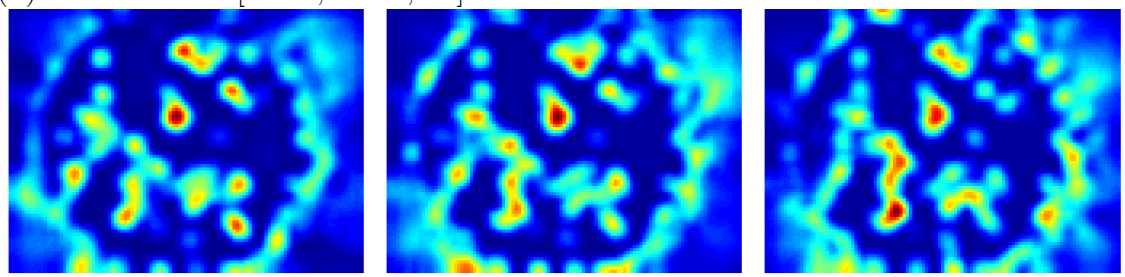
Fig. 4.6: **Experimental Data: Improvement of Best Solution:** Errors of the best solution at each generation as the population evolves. The best solution is Pareto optimal with the lowest product of objectives, but all the errors are shown separately. Data are the average from the four repeats of $abr\beta_1$.

(a) Target Colony $abr\beta_1$



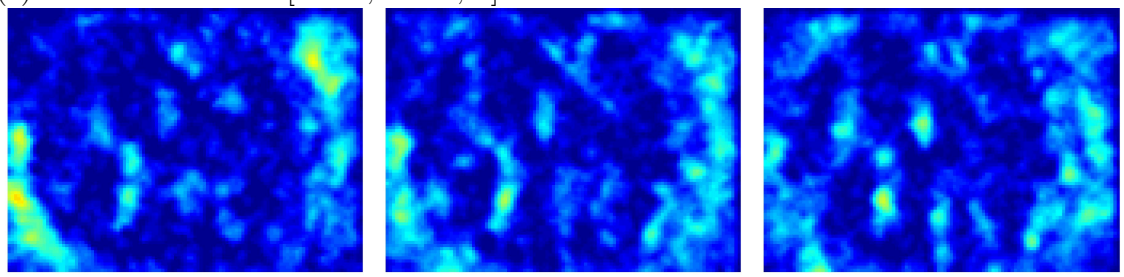
(a) Iterations 3-8 (b) 8-13 (c) 13-18

(b) Best solution [0.057, 0.155, 12]



(d) Iterations 3-8 (e) 8-13 (f) 13-18

(c) Random solution [1.563, 0.002, 7]



(g) Iterations 3-8 (h) 8-13 (i) 13-18

Fig. 4.7: **Experimental Data: Convergence Feature:** The convergence feature of the $\Delta abr\beta_1$ colony and example solutions from a fitting run of the search algorithm. Each image shows the combined convergence for the real data (a) and the simulation (b and c) at the beginning (iteration 3 - 8), middle (iteration 9 - 12), and end (iterations 13 - 18). (a) The experimentally measured convergence values extracted from the florescent bead movements within the developing colony. (b) The convergence values for the best solution found by the algorithm with parameter vector [.091, 1.974, 8]. Note the weaker correspondence (as compared to that of the synthetic target and best solution shown in section 4.1.1) between the experimental data and the best solution found. (c) Convergence for the three time periods from an initial random solution with parameter vector [.723, 2.882, 5]. Note the complete lack of correspondence between these convergence values and the target. The best solution is Pareto optimal with the lowest product of the three objectives. All images use the same color scale with red being high convergence and blue being low convergence.

for model accuracy. With inaccurate parameter values, the model cannot be used to gain understanding of the real biological system nor can it be used to make accurate predic-tions. The multicellular, spatiotemporal behavior of a model can be compared to observed behavior on the basis of multicellular features. Features are extracted from measurements of multicellular behavior relevant to the biological system under study. A feature is an abstraction of a specific measurement that extracts meaningful information which helps to constrain the parameter search. Additionally, a feature with the same semantics must be written to extract similar information from the simulated morphologies.

A robust parameter fitting software system was developed, validated, and applied. The software has potential in fitting the parameters of a wide array of multicellular models. It fits model parameters by extracting and comparing simulation features to features derived from supplied experimental data. It utilizes a parallel and distributed multi-objective ge-netic algorithm. The algorithm runs dozens to hundreds of simulations simultaneously and semi-synchronously to evolve the current generation of solutions to the next generation via crossover, mutation and selection. Multi-objective genetic algorithms in general are well-suited to find good solutions for difficult optimization problems such as those with a solution space that is potentially non-convex, multi-modal, non differentiable, and dis-continuous. Furthermore, the use of Pareto-optimality facilitates fitting on the basis of multiple, possibly orthogonal objectives. Together, these strengths of the developed search algorithm make it ideal in fitting multicellular models with a variety of solution spaces.

A multitude of specific genetic algorithms exist, many of which have been implemented within the software system utilized here [26]. In this case the NSGA-II algorithm [27] was chosen because it is effective in multi-objective optimization problems. To improve efficiency, the software was extended so that multiple solution evaluations and the entailed simulation runs could be performed in parallel. This necessitated that the evolution of the population of solutions become asynchronous. Each simulation of the biofilm model is single-threaded, which means the upper limit for the number of parallel threads is the same as the population size of a single generation. Given that each simulation takes hours,

further improvements will require parallelization of the simulation software.

The biofilm model represents a group of adjacent cells and the surrounding ECM by a spherical particle with a specific 3D location and volume. Several hundred-thousand particles are initialized under compression, which puts them in a quasi-stable state and mimics the adhesion ECM provides a colony as it grows. Early on in a simulation, particles near the biofilm-substratum boundary are removed from the simulation in localized regions according to a 2D cell death pattern (CDP), derived from a pattern observed in a real colony. Cell death perturbs colony stability, causing it to buckle and fold over the regions of cell death, ultimately resulting in wrinkle formation. Spatiotemporal convergence, partitioned into three temporal chunks, was used as the metric for evaluating a solution's fitness.

The algorithm was successfully validated with synthetic data in that it was able to reliably converge to values near to the parameters of several different target simulations. The algorithm regularly found solutions with low objective error and parameters closer to the synthetic target as the population evolved through its generations, illustrated in Figure 4.2. Interestingly, in the summary Table 4.1, discrepancies in the values for stiffness s_b and time S_i are low. However, scant consistency is exhibited in the values of ss_b, leading to a high standard-deviation and high error. This observation suggests that the model is insensitive to ss_b since a large range of values produce high quality fits to the experiments. Reviewing the real experimental error results, found in Table 4.2, we see that the same pattern exists, in that values for ss_b have high standard deviation over multiple genetic algorithm runs. In this case, results from fitting synthetic data accurately predicted that bonding force between the real biofilm and the stratum may not be critical to final morphology. This could be confirmed by further controlled experiments with changing stratum characteristics. Predictions of the sensitivity of model parameters suggests that initial studies with model simulations may help guide parameter selection and even guide which experimental observations are likely to be informative.

Following the synthetic studies, the algorithm was applied to fitting model parame-ters using data derived from experimental observations of two strains of *Bacillus subtilis*:

wild type (WT) and a genetic variant ($\Delta abr\beta$). It has been experimentally observed that compared to wild type, $\Delta abr\beta$ colonies are stiffer and wrinkle more slowly. The search found distinct sets of model parameter values for these two strains, which qualitatively replicate the aforementioned experimental observations. In particular, search consistently found higher particle-particle and particle-boundary stiffness for the $\Delta abr\beta$ colonies than it did for WT the colonies. Further, it found a higher shoving number for WT than for $\Delta abr\beta$. Intuitively, resolving forces and moving them (this is controlled by the number of shoves) more often in an iteration should result in more particle movement in an iteration and thus faster wrinkling. Real WT colonies wrinkle faster than $\Delta abr\beta$, hence the higher shove number found for the former. Thus, together the stiffness parameters and the shoving parameter control wrinkle formation rate, which allowed the search to find values for them that replicate the distinct wrinkling rates of the two strains.

None of the searches for parameter values in the real biofilms reduced the objective error to zero, see Table 4.2. This error results from the lack of correspondence between the observed and simulated convergence values as illustrated in Figure 4.7. In addition, note that the objective error illustrated in Figure 4.6 is higher at the end of the simulation (iterations 14 - 18) compared to the beginning and middle, suggesting that discrepancies due to model deficiencies increase as the simulation progresses. The presence of this residual error is significant since it implies that the model itself is lacking important biochemical and biomechanical processes, as well as cellular mechanisms. Just as in the prediction of low-influence mechanisms (biofilm/stratum bonding) described above, this residual error may be exploited to seek additional embellishments to the model. Correct model extensions are expected to reduce the residual error. Experiments may be suggested that manipulate and observe these proposed additional mechanisms.

The mechanical model used in this work is simple and facilitates simulations whose run time is adequate, but because many thousands of simulations are needed for large scale search, the limits must be made of run time. Currently, the wrinkling of biofilms with 10^6 particles can be simulated in a few hours on a modern personal computer. However, some

important aspects of biofilm development and wrinkle formation were omitted for simplicity and to avoid increasing run times. The model contains no representation of diffusible solutes like nutrients and waste nor sub-cellular processes. Simulations are initialized as if they have already grown for 36 hours under ECM-induced compression. Cell death is modeled by removing cells in the observed pattern and all-at-once, thus it does not occur emergently. Even with these limitations we believe the model captures the most important mechanical aspects of the folding and wrinkling that occurs after heterogeneous cell-death.

The available experimental data was limited. Observations of the spatiotemporal wrinkle height and wavelength spectrum would likely give the search process more information in order to find better fitting parameters. By introducing more error objectives that must be minimized, the parameter space becomes more constrained, opening up the possibility of increasing the number of parameters and thus incorporating important missing biochemical processes.

We intend to address these limitations in future work. We are currently in the process of developing and expanding the biofilm model using a more robust, highly parallel software package written in C++ called Biocellion [18] that runs on the cloud. This will allow us to include realistic cell growth and death and explicit ECM representation by tying their dynamics to diffusible solutes and sub-cellular processes via gene regulatory networks without compromising fast execution time. Further, we plan on applying the search algorithm to fitting this expanded model, which will include many more parameters. The limiting factor will then become experimental data.

This method of exploiting experimental data at the macro scale (multicellular outcomes) to fit parameters at the meso scale (cellular mechanisms) has great potential to increase the fidelity of multiscale models. Increased fidelity will lead to better prediction and a clearer scientific understanding of the cellular behaviors at work in these complex biological systems. Moreover, simulation and automated fitting can lead to insights into those cellular mechanisms that are most influential and help identify the need for model extensions. Both these outcomes can help focus directed experimentation and observation,

and ultimately improve the efficiency of the cycle between experimentation and modeling.

REFERENCES

[1] G. M. Whitesides, "Self-Assembly at All Scales," *Science*, vol. 295, no. 5564, pp. 2418–2421, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1126/science.1070821

[2] C. Woodford and P. W. Zandstra, "Tissue engineering 2.0: guiding self-organization during pluripotent stem cell differentiation." *Current opinion in biotechnology*, vol. 23, no. 5, pp. 810–819, Oct. 2012. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/22444525

[3] M. Asally, M. Kittisopikul, P. Rué, Y. Du, Z. Hu, T. Çağatay, A. B. Robinson, H. Lu, J. Garcia-Ojalvo, and G. M. Süel, "Localized cell death focuses mechanical forces during 3D patterning in a biofilm." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 109, no. 46, pp. 18 891–18 896, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1073/pnas.1212429109

[4] M. Trejo, C. Douarche, V. Bailleux, C. Poulard, S. Mariot, C. Regeard, and E. Raspaud, "Elasticity and wrinkled morphology of Bacillus subtilis pellicles," *Proceedings of the National Academy of Sciences*, vol. 110, no. 6, pp. 2011–2016, Feb. 2013. [Online]. Available: http://dx.doi.org/10.1073/pnas.1217178110

[5] G. H. Markx, J. S. Andrews, and V. P. Mason, "Towards microbial tissue engineering?" *Trends in biotechnology*, vol. 22, no. 8, pp. 417–422, Aug. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.tibtech.2004.06.009

[6] H. Vlamakis, C. Aguilar, R. Losick, and R. Kolter, "Control of cell fate by the formation of an architecturally complex bacterial community." *Genes & development*, vol. 22, no. 7, pp. 945–953, Apr. 2008. [Online]. Available: http://dx.doi.org/10.1101/gad.1645008

[7] T. Mammoto and D. E. Ingber, "Mechanical control of tissue and organ development." *Development (Cambridge, England)*, vol. 137, no. 9, pp. 1407–1420, May 2010. [Online]. Available: http://dx.doi.org/10.1242/dev.024166

[8] L. A. Lardon, B. V. Merkey, S. Martins, A. Dötsch, C. Picioreanu, J.-U. U. Kreft, and B. F. Smets, "iDynoMiCS: next-generation individual-based modelling of biofilms." *Environmental microbiology*, vol. 13, no. 9, pp. 2416–2434, Sep. 2011. [Online]. Available: http://dx.doi.org/10.1111/j.1462-2920.2011.02414.x

[9] X. Wang, M. Hao, and G. Wang, "Numerical simulation of wrinkle morphology formation and the evolution of different Bacillus subtilis biofilms." *Water science and technology : a journal of the International Association on Water Pollution Research*, vol. 73, no. 3, pp. 527–534, 2016. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/26877034

[10] W. Materi and D. S. Wishart, "Computational systems biology in drug discovery and development: methods and applications." *Drug discovery today*, vol. 12, no. 7-8, pp. 295–303, Apr. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.drudis.2007.02.013

[11] P. Vicini, "Multiscale modeling in drug discovery and development: future opportunities and present challenges." *Clinical pharmacology and therapeutics*, vol. 88, no. 1, pp. 126–129, Jul. 2010. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/20520608

[12] A. W. Mahoney, G. J. Podgorski, and N. S. Flann, "Multiobjective optimization based-approach for discovering novel cancer therapies." *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, vol. 9, no. 1, pp. 169–184, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1109/tcbb.2010.39

[13] J. G. Diaz Ochoa, J. Bucher, A. R. Péry, J. M. Zaldivar Comenges, J. Niklas, and K. Mauch, "A multi-scale modeling framework for individualized, spatiotemporal

prediction of drug effects and toxicological risk." *Frontiers in pharmacology*, vol. 3, 2012. [Online]. Available: http://dx.doi.org/10.3389/fphar.2012.00204

[14] J. A. Lerman, D. R. Hyduke, H. Latif, V. A. Portnoy, N. E. Lewis, J. D. Orth,

A. C. Schrimpe-Rutledge, R. D. Smith, J. N. Adkins, K. Zengler, and B. O. Palsson, "In silico method for modelling metabolism and gene product expression at genome scale." *Nature communications*, vol. 3, pp. 929+, Jul. 2012. [Online]. Available: http://dx.doi.org/10.1038/ncomms1928

[15] D. Murray, P. Doran, P. MacMathuna, and A. C. Moss, "In silico gene expression analysis–an overview." *Molecular cancer*, vol. 6, no. 1, pp. 50+, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1186/1476-4598-6-50

[16] A. V. Ratushny, S. A. Ramsey, O. Roda, Y. Wan, J. J. Smith, and J. D. Aitchison, "Control of transcriptional variability by overlapping feed-forward regulatory motifs." *Biophysical journal*, vol. 95, no. 8, pp. 3715–3723, Oct. 2008. [Online]. Available: http://dx.doi.org/10.1529/biophysj.108.134064

[17] T. Kim, K. A. Afonin, M. Viard, A. Y. Koyfman, S. Sparks, E. Heldman, S. Grinberg, C. Linder, R. P. Blumenthal, and B. A. Shapiro, "In Silico, In Vitro, and In Vivo Studies Indicate the Potential Use of Bolaamphiphiles for Therapeutic siRNAs Delivery." *Molecular therapy. Nucleic acids*, vol. 2, 2013. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/23511334

[18] S. Kang, S. Kahan, J. McDermott, N. Flann, and I. Shmulevich, "Biocellion: accelerating computer simulation of multicellular biological system models," *Bioinformatics*, vol. 30, no. 21, pp. 3101–3108, Nov. 2014. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btu498

[19] P. V. Liedekerke, M. M. Palm, N. Jagiella, and D. Drasdo, "Simulating tissue mechanics with agent-based models: concepts, perspectives and some novel results," *Computational Particle Mechanics*, vol. 2, no. 4, pp. 401–444, Dec. 2015. [Online]. Available: http://dx.doi.org/10.1007/s40571-015-0082-3

[20] F. Castiglione, F. Pappalardo, C. Bianca, G. Russo, and S. Motta, "Modeling Biology Spanning Different Scales: An Open Challenge," *BioMed Research International*, vol. 2014, pp. 1–9, 2014. [Online]. Available: http://dx.doi.org/10.1155/2014/902545

[21] M. K. Transtrum, B. B. Machta, and J. P. Sethna, "Why are nonlinear fits so challenging?" *Physical Review Letters*, vol. 104, no. 6, pp. 060 201+, Dec. 2009.[Online]. Available: http://dx.doi.org/10.1103/physrevlett.104.060201

[22] Q. B. Baker, G. J. Podgórski, C. D. Johnson, E. Vargis, and N. S. Flann, "Bridging the multiscale gap: Identifying cellular parameters from multicellular data," in *Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2015 IEEE Conference on.* IEEE, Aug. 2015, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/cibcb.2015.7300323

[23] E. Balsa-Canto, M. Peifer, J. R. Banga, J. Timmer, and C. Fleck, "Hybrid optimization method with general switching strategy for parameter estimation." *BMC systems biology*, vol. 2, no. 1, pp. 26+, Mar. 2008. [Online]. Available: http://dx.doi.org/10.1186/1752-0509-2-26

[24] G. Lillacci and M. Khammash, "Parameter Estimation and Model Selection in Computational Biology," *PLoS Comput Biol*, vol. 6, no. 3, pp. e1 000 696+, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1371/journal.pcbi.1000696

[25] J. Sun, J. M. Garibaldi, and C. Hodgman, "Parameter Estimation Using Metaheuristics in Systems Biology: A Comprehensive Review," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 1, pp. 185–202, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1109/tcbb.2011.63

[26] D. Hadka, "MOEA Framework Software Package and User Guide, available from http://www.moeaframework.org/," 2015. [Online]. Available: http://www. moeaframework.org

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1109/4235.996017

[28] S. Kukkonen and J. Lampinen, "GDE3: the third evolution step of generalized differential evolution," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2005, pp. 443–450 Vol.1. [Online]. Available: http://dx.doi.org/10.1109/cec.2005.1554717

[29] E. Zitzler and S. KÃnzli, "Indicator-based selection in multiobjective search," in *in Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII*, 2004, pp. 832–842. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.1604

[30] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, and J. Martin, "PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCOâ2001*, 2001, pp. 283–290. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.2194

[31] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," pp. 95–100, 2001. [Online]. Available: http: //citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.4617

[32] J. D. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms.* Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 93–100. [Online]. Available: http://portal.acm.org/citation.cfm?id=657079

[33] M. Marvasi, P. T. Visscher, and L. Casillas Martinez, "Exopolymeric substances (EPS) from Bacillus subtilis: polymers and genes encoding their synthesis." *FEMS*

microbiology letters, vol. 313, no. 1, pp. 1–9, Dec. 2010. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/20735481

[34] H.-C. Flemming and J. Wingender, "The biofilm matrix," *Nat Rev Micro*, vol. 8, no. 9, pp. 623–633, Sep. 2010. [Online]. Available: http://dx.doi.org/10.1038/nrmicro2415

[35] D. Schultz, J. N. Onuchic, and E. Ben-Jacob, "Turning death into creative force during biofilm engineering," *Proceedings of the National Academy of Sciences*, vol. 109, no. 46, pp. 18 633–18 634, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1073/pnas.1215227109

[36] J. S. Webb, M. Givskov, and S. Kjelleberg, "Bacterial biofilms: prokaryotic adventures in multicellularity." *Current opinion in microbiology*, vol. 6, no. 6, pp. 578–585, Dec. 2003. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/14662353

[37] D. B. Kearns, F. Chu, S. S. Branda, R. Kolter, and R. Losick, "A master regulator for biofilm formation by Bacillus subtilis." *Molecular microbiology*, vol. 55, no. 3, pp. 739–749, Feb. 2005. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/15661000

[38] J. N. Wilking, V. Zaburdaev, M. De Volder, R. Losick, M. P. Brenner, and D. A. Weitz, "Liquid transport facilitated by channels in Bacillus subtilis biofilms." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 110, no. 3, pp. 848–852, Jan. 2013. [Online]. Available: http://dx.doi.org/10.1073/pnas.1216376110

[39] L. E. Dietrich, C. Okegbe, A. Price-Whelan, H. Sakhtah, R. C. Hunter, and D. K. Newman, "Bacterial community morphogenesis is intimately linked to the intracellular redox state." *Journal of bacteriology*, vol. 195, no. 7, pp. 1371–1380, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1128/jb.02273-12

[40] M. A. Hamon, N. R. Stanley, R. A. Britton, A. D. Grossman, and B. A. Lazazzera, "Identification of AbrB-regulated genes involved in biofilm formation by Bacillus subtilis." *Molecular microbiology*, vol. 52, no. 3, pp. 847–860, May 2004. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/15101989

[41] S. S. Branda, J. E. González-Pastor, S. Ben-Yehuda, R. Losick, and R. Kolter, "Fruiting body formation by Bacillus subtilis." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 20, pp. 11 621–11 626, Sep. 2001. [Online]. Available: http://dx.doi.org/10.1073/pnas.191384198

[42] D. Romero, C. Aguilar, R. Losick, and R. Kolter, "Amyloid fibers provide structural integrity to Bacillus subtilis biofilms." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, no. 5, pp. 2230–2234, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1073/pnas.0910560107

[43] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.ress.2005.11.018

[44] I. Diakonikolas, "Approximation of Multiobjective Optimization Problems," Ph.D. dissertation, Columbia University, 2011. [Online]. Available: http://hdl.handle.net/10022/AC:P:20500

[45] M. Caramia and P. Dell'Olmo, *Multi-objective Optimization*. Springer-Verlag, 2008, ch. 2, pp. 11–36. [Online]. Available: http://www.springer.com/us/book/9781848003811

[46] B. Suman and P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization," *Journal of the Operational Research Society*, vol. 57, no. 10, pp. 1143–1160, Oct. 2006. [Online]. Available: http://dx.doi.org/10.1057/palgrave.jors.2602068

[47] T. Jones and S. Forrest, "Genetic Algorithms and Heuristic Search," in *Santa Fe Institute*, 1995, pp. 16–20. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.5925

[48] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=559152

[49] P. of Vision Pty. Ltd., "Persistence of vision raytracer (version 3.6)," http://www.povray.org/download/, 2004.