Distributed Computing in Nature

Sabrina Rashid

CMU-CB-19-100

April 2019

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Ziv Bar-Joseph, Chair
Russell Schwartz
Hanna Salman
Saket Navlakha

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

The work was supported by the National Heart, Lung, and Blood Institute of the National Institutes of Health under award number U01HL122626 and by the National Science Foundation under Grant No. DBI-1356505.

The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health or the National Science Foundation.

Keywords: Bacteria Swarm, Histone modification, Shared Memory, Distributed Gradient Descent

To Imrul

Abstract

There are many parallel requirements of computational and biological systems, suggesting that each can learn from the other. Like virtually all large-scale computing platforms, biological systems are mostly distributed consisting of molecules, cells, or organisms that interact, coordinate, and reach decisions without central control. However, unlike most computational methods, biological systems rely on very limited communication protocols, do not assume that the identity of the communicating agents is known and only utilize simple computations. This makes biological systems more robust to interference and environmental noise, and allows them to function efficiently in harsh settings, a property that is desirable in computational systems as well. Recent work demonstrated that for certain network based distributed algorithms, our improved ability to study biological processes at the molecular and cellular levels allows us to both improve our understanding of these biological processes as well as suggest novel ways to improve distributed computational algorithms. In this thesis we study distributed computing models for two natural phenomena, i) food search in bacteria swarm, ii) post translational modification in epigenetics. For bacteria, we propose a model based on an extension of the distributed gradient descent (DGD) algorithm. We show that our DGD model accurately predicts the true dynamics of bacterial chemotaxis while raising a number of novel hypotheses, some of which we experimentally validated. We then show that the revised DGD model can be used to improve the performance of robotic swarms and for efficient response during mass evacuations. For epigenetics, we present a distributed shared memory consensus model for explaining post translational histone modifications. To our knowledge this is the first time that a shared memory model (rather than message passing) is used for studying a biological coordination process.

Acknowledgments

First, I would like to express sincere gratitude towards my supervisor Prof. Ziv Bar-Joseph for his continuous support and advice. His enthusiasm kept me motivated through out the thesis. My PhD has been shaped positively and fundamentally by his patient and kind guidance. Ziv has the inexplicable ability to push me to heights I never knew were possible to achieve, and without him the contributions of this thesis would have fallen much further short of what I was unknowingly capable.

Experimental validation is a key part of this thesis. There are several people without whom this would not have been possible. To that end, first I would like to thank our collaborators Drs. Zoltan Oltvai and Hanna Salman for conceiving and facilitating the experiments on the bacteria swarm project. Their feedback and biological insight have been very crucial in my computational model development as well. I would like to thank Dr. Zhicheng Long for designing and conducting several experiments on bacteria swarm migration. His experiments are an indispensable part of the thesis. I would also like to thank Maryam Kohram and Harsh Vashistha from Dr. Salman's group for their help in conducting additional experiments and analyzing the data.

Next, I would like to thank Dr. Gadi Taubenfeld for his valuable contribution towards the DNA shared memory project. Without his active collaboration on the theoretical analysis of the model, the project would have been incomplete. I am thankful to Dr. Anupam Gupta as well for his contribution towards the theoretical analysis of the shared memory model.

I am immensely grateful for the opportunity to work in the Bar-Joseph group. I met and collaborated with a amazing group of people with great intellect, who went above and beyond to help me with not only the thesis but also coping up with the strenuous graduate student life. I am also very thankful to the fellow CPCB students of my class. From the very beginning of my PhD, I leaned on them and received unwavering support both personally and academically. I will always cherish their friendship.

I would like to thank my amazing parents back in my country Bangladesh, my mother Lutfa Sultana, father Abdur Rashid Dewan. It is difficult being so far away but their strong belief in me have always kept me motivated to achieve my goal. I would also like to take the opportunity to thank my brother Toufiqul Islam for his continuous advice and support.

Lastly, I would like to thank my dear husband Imrul. He has been with me every step of the way from the beginning to the end. Without his loving support this PhD would not have been possible.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
  - 1.1.1 Examples of distributed biological algorithms . . . 2
- 1.2 Bacteria chemotaxis . . . 4
  - 1.2.1 Bacterial chemotaxis model for mass evacuation . . . 6
- 1.3 Genomewide epigenetic modification in DNA . . . 8
  - 1.3.1 Shared memory models in distributed computing . . . 11
- 1.4 Thesis organization . . . 12

2 Distributed Gradient Descent in Bacterial Food Search **13**
- 2.1 Introduction . . . 13
- 2.2 Methods . . . 14
  - 2.2.1 A computational framework for understanding bacterial food search . . . 14
  - 2.2.2 The Shklarsh model: Cell based differential equation computation . . . 16
  - 2.2.3 A distributed gradient descent model . . . 18
  - 2.2.4 Proof of convergence . . . 21
  - 2.2.5 Terrain modeling . . . 28
- 2.3 Results . . . 28
  - 2.3.1 Simulation parameters . . . 29
  - 2.3.2 Performance on a realistic food search simulation . . . 29
  - 2.3.3 Sensitivity to silent agents . . . 33
  - 2.3.4 Effect of different modeling components . . . 36
- 2.4 Discussion . . . 39

3 A Bacterial based Distributed Gradient Descent Model for Mass Scale Evacuations **43**
- 3.1 Introduction . . . 43
- 3.2 Method . . . 45
  - 3.2.1 Computational model . . . 45
  - 3.2.2 Individual environment map and gradient . . . 46
  - 3.2.3 Communication between agents . . . 47
- 3.3 Experimental results . . . 49
  - 3.3.1 Basic room with varying door size and obstacles . . . 52
  - 3.3.2 Real floor plan of a building . . . 53
  - 3.3.3 Using Euclidean distance instead of geodesic distance . . . 59

3.3.4 Sensitivity analysis of Bacterial DGD hyperparameters 60
3.4 Discussion 61

4 Adaptive Tumbling in Bacterial Chemotaxis on Obstacle-laden Terrains 65
4.1 Introduction 65
4.2 Computational methods 66
4.2.1 Mathematical and computational models for bacterial chemotaxis 66
4.2.2 DGD model for bacterial chemotaxis 68
4.2.3 Revised DGD model 70
4.3 Experimental methods 73
4.3.1 Bacterial strains and growth conditions 73
4.3.2 Microfluidic device 73
4.3.3 Time-lapse imaging 74
4.4 Analyzing single cell tracks 75
4.4.1 Computing tumbling rates 79
4.4.2 Mean Square Displacement (MSD) analysis 79
4.5 Results 81
4.5.1 Analysis of existing chemotaxis models 81
4.5.2 Experimental results contradict model predictions 81
4.5.3 Performance of revised DGD model 84
4.5.4 Experimentally testing model predictions 88
4.5.5 Sensitivity analysis of the parameters of revised DGD model 94
4.5.6 Testing the revised DGD model on a computational system 99
4.6 Discussion 102

5 Genome Wide Epigenetic Modifications as a Shared Memory Consensus Problem 107
5.1 Introduction 107
5.2 Model 108
5.3 The consensus write-erase problem 110
5.4 Algorithms and analysis 111
5.4.1 A naive algorithm 111
5.4.2 The consensus write-erase algorithm 111
5.4.3 Reaching consensus for biased coin flips 112
5.4.4 Runtime analysis for biased coin flipping 113
5.4.5 Using the analysis for the coin-flip consensus to analyze the write-erase consensus 114
5.4.6 Resolving multiple collisions in parallel 115
5.5 Simulations and analysis of real biological data 116
5.6 Discussion 119

6 Conclusion and Future Work 123
6.1 Summary of contribution 123
6.2 Future work 125
6.2.1 Bacterial chemotaxis and DGD 125

6.2.2 Shared memory models 127
6.2.3 Other biologically distributed algorithms 127

Bibliography **129**

List of Figures

1.1 Examples of biological-computational bi-directional studies . . . 3
1.2 Bacteria chemotaxis . . . 4
1.3 Emergency evacuation of a office layout. . . . 7
1.4 Structural organization of a chromosome . . . 9
1.5 Chromatin states . . . 10

2.1 Bacterial food search and dynamics of cell-cell communication in DE model . . . 15
2.2 Dynamics of cell-cell communication in DGD model . . . 20
2.3 Comparison of search time under different communication models . . . 30
2.4 Performance of multiple groups . . . 30
2.5 Effect of dynamic obstacles . . . 32
2.6 Effect of different weighting functions . . . 34
2.7 Effect of silent agents . . . 35
2.8 Effect of agents sending wrong messages . . . 36
2.9 Effect of different communication components . . . 37
2.10 Effect of weighted communication . . . 38
2.11 Effect of population sizes . . . 39

3.1 Gaussian cost function of a real floor plan with one exit . . . 50
3.2 Simulations of a basic rectangular room. . . . 51
3.3 Simulation with the real floor plan . . . 54
3.4 Performance with individual sensing only . . . 56
3.5 Comparison of models using full communication . . . 57
3.6 Effect of repulsion . . . 58
3.7 Performance with multiple obstacles . . . 59
3.8 Comparison between Euclidean and geodesic distance metric . . . 60

4.1 Layout of original and revised DGD model . . . 71
4.2 Simulation results of current chemotaxis models . . . 76
4.3 Different bacteria migration chemotaxis model performance under varying obstacle sizes . . . 77
4.4 Simulation results of the revised bacterial DGD model and prior chemotaxis models with approximately 20% immobile cells . . . 78
4.5 Experimental results and revised DGD model . . . 82
4.6 Different microchamber setups . . . 83

4.7 Experimental results and revised DGD model for cells in 200μM aspartate (Asp) 85
4.8 Gradient profile in simulation in the presence of obstacles 86
4.9 Experimental results and revised DGD model with round obstacles 86
4.10 Experimental results and revised DGD model with round obstacles 87
4.11 Performance of updated Shklarsh model 89
4.12 Single cell trajectories under different obstacle coverages 91
4.13 Distribution of tumbling frequency under different obstacle coverages 92
4.14 Distribution of tumbling frequency under different obstacle coverages in simulation 93
4.15 Cell tracks under 64% obstacle coverage simulations 94
4.16 Log-log plot of MSD vs time (s) under different obstacle coverages 95
4.17 Irregular obstacle setup 96
4.18 Distribution of bacterial chemotactic food search time in variable obstacle setups 97
4.19 Application of adaptive tumbling frequency method for swarm robot search task . 100
4.20 Simulation parameters used on NetLogo software. 101

5.1 Biased coin flipping experiment 113
5.2 Distribution of number of steps to reach consensus 116
5.3 Number of zeros and collisions vs steps in the proposed algorithm 117
5.4 Tracks of competing histone modifiers 120
5.5 Consensus of histone marks 121

List of Tables

3.1 Sensitivity analysis of Bacterial DGD hyperparameters 62

4.1 Sensitivity analysis of revised DGD model parameters 98

Chapter 1

Introduction

1.1 Background

Several autonomous computational systems, including robotic swarms and sensor networks, are expected to communicate, compute and coordinate without central control [1, 2]. A number of machine learning methods have either been developed or adopted for these tasks including belief propagation and singular value decomposition [3, 4]. These methods work by integrating the internal knowledge or belief of an individual agent with a weighted function of the belief of its neighbors to determine the next action the agent should take. While such methods have been successfully used in several applications, they also suffer from some drawbacks that limit their usability in constrained settings. For example, most methods assume a static interaction network, whereas in most realistic settings the set of neighbors and the environment is constantly changing [5]. In addition, current algorithms rely on the identifiability of neighboring agents, their ability to send and receive large messages and to compute complex functions. These requirements can consume significant energy and thus may reduce the ability of distributed methods to perform well in large, ad-hoc and congested settings.

Like distributed computational systems, most biological processes also operate in a distributed manner via networks of molecules, cells, or organisms. These biological systems, even

though limited in computation and communication ability compared to computational systems, often perform very robustly and efficiently in adverse environment. These properties are very advantageous in computational systems as well, i.e., performing efficiently and robustly while consuming minimum resources. Hence by studying such distributed biological systems we can propose new or improve existing computational algorithms and at the same time improve the understanding of the biological systems. For example, a number of key machine learning optimization algorithms, including neural networks and non-negative matrix factorization, have been inspired by information processing in the brain [6], [7].

Recently, several studies on how distributed biological systems process information and solve computational problems have been published [8], [9], [10]. Our ability to study the inner mechanisms of molecular and cellular systems have majorly contributed to this increased interest lately. As mentioned above, widescale use of distributed computing in wireless, mobile, and sensor device has also called for more robust framework for distributed computing. Together, the new data and communication models present a unique opportunity to jointly model, analyze, and learn from biological systems.

1.1.1 Examples of distributed biological algorithms

To date, there have been several studies on computational modeling of distributed biological systems. For example, studies have demonstrated that ants solve the foraging problem by implementing a version of the Transmission Control Protocol (TCP), which is used on the Internet to determine available bandwidth when routing packets [8]. Despite being limited in communication (either physical contacts or pheromone trails) and computation power, the ants foraging scheme is very robust to environmental factors. We also see parallels of distributed computing in probabilistic firing of neurons. It has been experimentally shown that neurons are often probabilistic, there is some randomness in neural output under the same input conditions [11]. Networks of such randomly activated neurons conduct probabilistic inference in a manner sim-

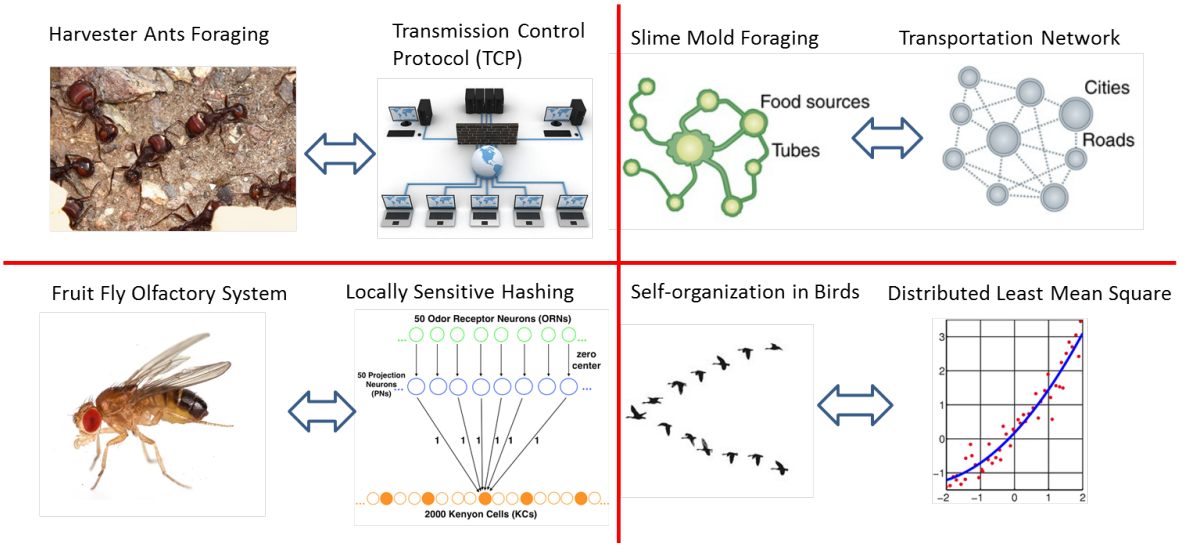

Figure 1.1: Examples of biological-computational bi-directional studies

ilar to Markov chain Monte Carlo sampling in distributed systems [9]. Recent study on fruit fly olfactory circuits show that fruit fly actually recognizes odors by implementing a version of locally sensitive hashing that is efficient and robust to environmental noise [12]. We also see similar distributed computing in the group formation dynamics of bird flocks or fish schools [13]. Distributed algorithms inspired from slime molds have been used to design efficient traffic networks in cities [10]. All these examples show the advantage of such biological-computational bi-directional studies. Fig. 1.1 shows few examples of such studies.

The thesis extends these studies by focusing on two additional biological processes that, to date, were not modeled as distributed computing systems. The first is bacteria swarm chemotaxis which is one of the most well studied processes in experimental biology. The second is genomewide epigenetic modification of DNA which is a key component of gene regulatory mechanism. In the following sections we describe these processes in detail and discuss some of the prior models of these processes.

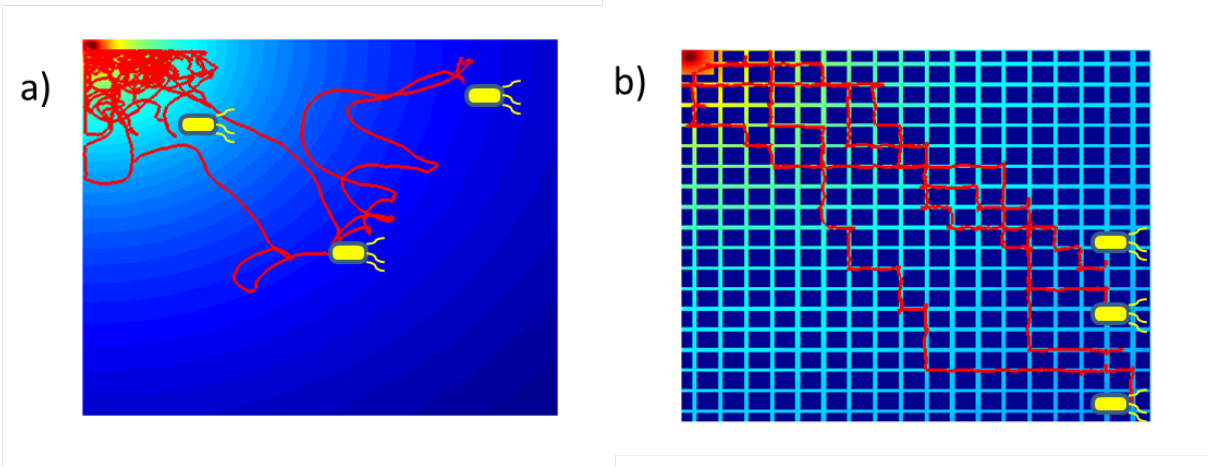

Figure 1.2: Bacteria chemotaxis in a) no-obstacle b) obstacle terrain. Bacteria cells follow the chemical gradient by a combination of swimming and tumbling movement. In each of the setups, three bacteria cell tracks following the chemical gradient is shown.

1.2 Bacteria chemotaxis

Chemotaxis refers to the movement of an organism in response to a chemical stimulus. Bacteria cells uses this technique to adapt to ever changing environmental conditions (Fig. 1.2). In addition to sensing the food source, the bacteria cells can also secrete additional chemicals in the environment that can be sensed by other cells. This works as a means of communication between the cells. Bacteria swarm utilizes this process to jointly perform a specific task (for example food search) much more efficiently than individual cells do [14] ,[15]. When searching for food in harsh environment, (e.g. obstacle laden terrain Fig. 1.2) such communication between the cells allows them to avoid the obstacles and find the food faster.

Detailed molecular studies have identified key proteins and signaling pathways involved in this process [15] (and references therein), [16], [17] while other studies have focused on the mode by which individual cells process information and secrete various signals [18] and on pairwise communication between cells [19]. Previous work has shown experimental evidence for cell-cell communication that significantly enhances the chemotactic migration of bacterial popu-

lations [20]. Specifically, it was shown that *E. coli* cells respond to a gradient of chemoattractant not only by biasing their own random-walk swimming pattern but also by actively secreting a strong chemoattractant into the extracellular medium that amplifies the migration of cells distant to the attractant source [20]. To gain insight into the ecosystem-level organization of chemotaxis, mathematical models have been developed to describe the collective behavior of cells [21, 22, 23, 24, 25] and simulations indicated that these capture several aspects of the observed behavior [26, 27]. One of the first models developed for bacterial chemotaxis is the Keller-Segel model [25] that describes the concentration profiles of bacteria and attractant over time using coupled differential equations. Later models were based on experiments from Brown and Berg et al. [28]. These models made the duration of bacterial trajectory following a tumbling step, at a specific direction, a function of the perceived attractant gradient [23, 24]. However, these studies did not account for the cooperative aspects of foraging bacteria. Current models that incorporate communication between cells [26],[29] are largely based on differential equation methods. While these can indeed lead to a fast, coordinated search, they do not fully take into account the dynamically changing topology of cells' interaction network over time. Furthermore, they make unrealistically strong assumptions about the abilities of cells to identify the source(s) of the messages and to utilize a large (effectively continuous valued) set of messages, which is unrealistic given the limited computational powers bacteria cells possess.

In this thesis, we introduce a distributed gradient descent (DGD) model that makes biologically realistic assumptions regarding the dynamics of the agents, the size of the set of the messages, and the agents' ability to identify senders. Classical distributed gradient descent is a first order iterative method originally developed by Nedic and Ozdaglar approximately a decade ago [30]. In each iteration of DGD, each node performs an update by using a linear combination of a gradient step with respect to its local objective function and a weighted average from its local neighbors (also termed as a consensus step). However in large networks, due to the large amount of data sharing and the communication bottleneck, exchanging full high-dimensional

information between neighboring nodes is time-consuming (or even infeasible), which significantly hinders the overall convergence of classical DGD. In this thesis, we propose a new version of the classical DGD algorithm that is consistent with bacterial communication and computation principles. The new version significantly simplifies the communication bottleneck of classical DGD. We show that our DGD model solves the bacterial food search problem more efficiently (in terms of the overall complexity of messages sent) and more quickly (in terms of the time it takes the swarm to reach the food source) when compared to current differential equation models and also leads to better agreement with experimental results.

While the DGD model explained and accurately predicted bacterial behavior, an important question remained about its potential application to distributed computational systems. Recent work has used other variants of DGD to improve the ability of various search algorithms to find an optimal distribution [31]. However, to date the use of the simplified, bacterial based, DGD for real communicating systems has not been studied. Here we study this issue and show that in dynamic coordination methods that can only use very limited resources DGD can provide advantages over state of the art methods.

1.2.1 Bacterial chemotaxis model for mass evacuation

In emergency evacuation scenarios (for example, following fire or explosion in a crowded location) individuals have very limited ability to communicate and the infrastructure maybe damaged leading to severe constraints on the ability to send, receive and process information (Fig. 1.3). In the context of computational modeling several prior studies have focused on such evacuations and these can be largely grouped based on their goals. The first group involves studies that attempt to accurately model human behavior so that simulations of mass scale evacuation can be performed when designing new buildings, stadiums, airports etc. These include modeling human behavior in crowded enclosed space [32], [33]. The second set of studies attempted to develop new algorithms for improving the coordination and the time for full evacuation regardless of the

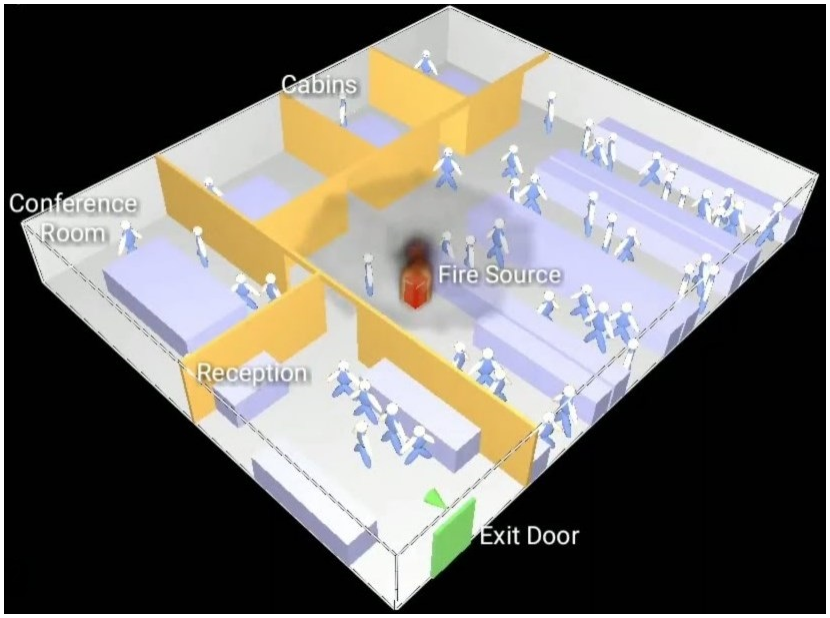

Figure 1.3: Emergency evacuation of a office layout.

setting in which it would be used. These methods included [34], [35], [36], [37]. Most of these methods in both groups are based on social force modeling (SFM) originally proposed in [32]. SFM use three components in their evacuation strategy, i) attraction force towards the exit, ii) repulsive force to agents that are closer than a certain distance threshold, and iii) repulsive/friction force from wall or obstacles. A variant of SFMs which weights the repulsion force based on the agent distances has also been studied [33], [38]. This variant accounts for some additional crowd psychological aspects, such as avoiding too crowded paths. However, while SFM is successful in certain settings, an important aspect of crowd behavior that is not addressed by SFM is the ability of agents to cooperate and communicate even under severe limitations which are likely to occur in such events.

To address this, a few recent methods were suggested for utilizing communication / cooperation among agents [34], [35]. These include guided evacuation via a leader [34] or by information transmission among the agents [35]. While the guided evacuation model may be helpful in some cases, it relies on strong assumptions about the existence of such leader and the ability of agents to identify it in extreme circumstances which is often not a realistic assumption. To address this,

in [35] the authors used mutual information between agents' position and velocity to adaptively modify some of the parameters of SFM (for example, collision radius). Another recently suggested evacuation method is based on selecting a single neighbor (based on some metric) and integrating the agents internal belief with that neighbor information in order to decide on the direction of the next movement [36], [37].

While the methods discussed above have been tested in a number of different settings, most of them were not tested in (and, in fact, were not developed for) dynamic environments in which new obstacles that are unknown a-priori to the agents are introduced (consider fire based evacuations). In such cases some passageway may become blocked preventing the agents from reaching certain exits. Following the previously known exit routes (either by individual choice or by following a leader or neighbor) will result in congestion at the blocked pathway and subsequently increase the evacuation time. None of the prior methods discuss how to optimize evacuations given such dynamic environmental changes.

To address these issues we developed an extension of the Bacterial DGD which maintains all of the positive aspects of the bacterial model including the reliance on simple messages, data aggregation and the ability to handle noisy environments, for use in mass evacuation situations. We show that in dynamic complex environments the DGD model achieves the best performance when compared to prior methods while still only using simple messages and computation, making it a viable alternative for such situations.

1.3 Genomewide epigenetic modification in DNA

Epigenetic refers in part to post translational modifications of the histone proteins on which the DNA is wrapped (Fig. 1.4). Stretches of about 150 base pairs are wrapped around octets of histone proteins to form nucleosomes [39]. These and other DNA-associated proteins make up *chromatin*. Researchers are currently cataloging chromatin proteins and their modifications, in an effort to segregate chromatin's complexity into discrete numbers of chromatin states [40].

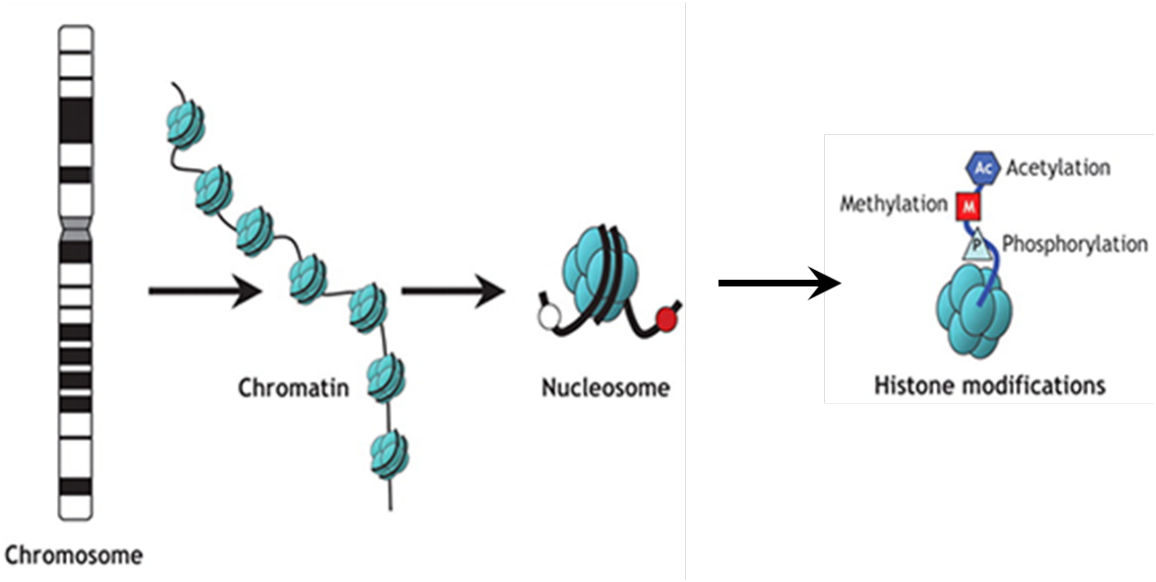

Figure 1.4: Structural organization of a chromosome. Stretches of DNA are wrapped around octets of histone proteins to form nucleosomes. Different types of modifications (e.g. acetylation, methylation, phosphorylation etc.) on the histone residues regulate chromatin states.

Chromatin-state mapping promises to reveal many secrets of genome function, how cells inherit acquired states, how chromatin directs functions such as transcription and RNA processing, and how chromatin biology contributes to disease.

Histone modifications play an important role in the regulation of chromatin states and so are themselves highly regulated and consistent across large stretches of the genome [41, 42]. In particular, when switching between cell states (for example, when facing stress or during development), a coordinated set of modifications are required such that expression programs required for these states can be executed. While exact time scales of such modification is still unknown,a few recent studies looked at time scales for writing histone marks in response to external stimulus [43]. For example, in [43] the authors studied the insertion of H3K9me3 after treatment with $csHP1\alpha$. They found that after around 18 hours H3K9me3 marks began appearing in the relevant segment, and within 5 days, a large domain of approximately 10 kbp of H3K9me3 had formed.

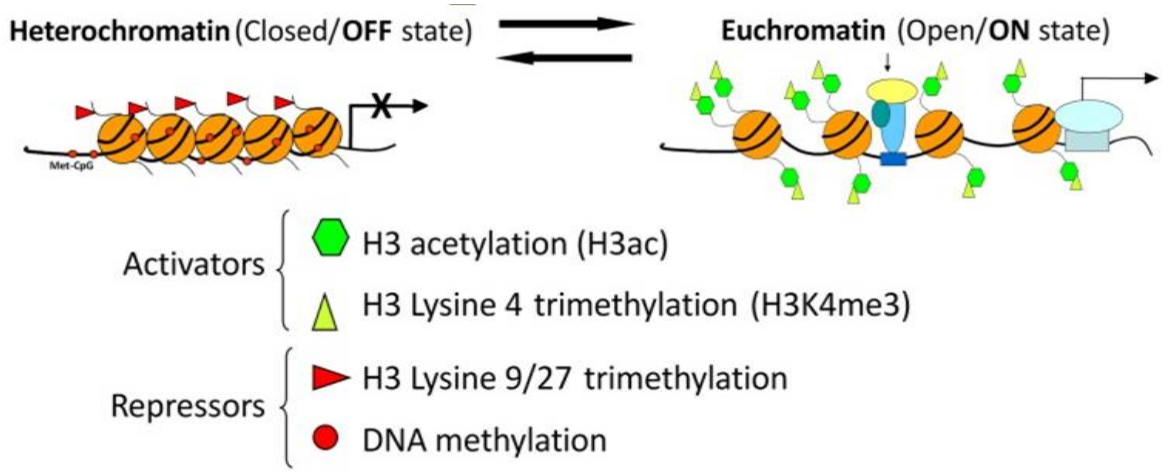

Figure 1.5: Different histone modifiers regulate the chromatin states (open/closed) by modifying a particular residue on the histones. Open chromatin state activates the gene by making in accessible for transcription. Similarly, closed chromatin states represses gene by making it inaccessible.

There are different types of histone modifiers that regulate transcriptional activity by changing the chromatin states (open/closed) (Fig. 1.5). These modifiers can be broadly categorized into three classes, Readers, Writers, and Erasures. Writers modify the histones that can translate to transcriptional activation or repression. Erasures can erase the marks originally put by the writers, so that new type of writer can write on that same histone location. Lastly readers see these marks and mediate the actions of writers and erasures to govern DNA transcription.

Modifiers are responsible for several types of modifications. These include acetylation, methylation, phosphorylation, ubiquitination, and so on. Each type of histone modification is related to a specific type of transcriptional regulation. For example, acetylation of Histone 3 lysine 9 (H3K9) residue can turn on genes, whereas methylation of the same residue can silence the genes (Fig. 1.5). It can also be noted from the above example that different types of modifications can happen at the same location of histone. But not at the same time, if a certain location is already written/marked/modified by a certain modifier, the mark has to be erased before it can be written by another modifier. Although a lot of recent work in the biological literature has

focused on histone modifications and the specific proteins that regulate this process, to the best of our knowledge no prior work discussed the issue of global coordination between writers and erasers. We often see *exactly the same* histone marks for a stretch of DNA [41]. However, how such consensus is reached for a particular mark is still not clear.

1.3.1 Shared memory models in distributed computing

While message passing is a useful and dominant method for distributed biological computing, for epigenetics, histone modification utilizes another method for coordination. This method is similar to the method of communicating via shared memory, studied intensively in distributed computing. In shared memory, multiple processes can read and write the same memory locations [44]. Shared memory models have been used to address important issues that arise in modern applications, including as fault-tolerance and availability of data centric services. Consensus is an important building block for realizing such reliable distributed systems [45]. In the histone modification paradigm, we can view histone modifiers (readers/writers/erasures) as processors and the DNA as the shared memory location. Different types of writers can write (put histone mark such as acetylation/methylation) on the DNA (shared memory), which can also be read/erased/re-written by other modifiers. Eventually stretches of DNA regions reach a consensus in terms of a particular histone mark. Even though the consensus is not global, the regions can occupy large portions of the DNA. Even though there are several types of histone modifiers, for simplicity of theoretical analysis we consider only two types of writer and eraser processors, and for such a setting formally define the *consensus write-erase problem*. We then present an algorithm for solving the problem that is consistent with various biological assumptions and discuss its run time both theoretically and in simulations.

1.4 Thesis organization

The thesis is organized as following. In Chapter 2, we formulate the DGD model for bacterial chemotactic food search. We discuss and compare the performance of the model with prior works under different simulation setups. We show that DGD model leads to shorter food search time while relying on biologically realistic assumption. We also show proof of convergence of the proposed DGD model given the dynamic interaction network of bacteria cells.

In chapter 3, we show the application of bacterial DGD model in designing evacuation strategy. We discuss the extension the bacterial DGD to incorporate human movement. We show that employing extended bacterial DGD as a evacuation strategy leads to efficient evacuation time in dynamic scenario, i.e., when unknown obstacles are present in the environment. We compared to several prior models of evacuation and show that unlike bacterial DGD these models significantly increase the evacuation time when faced with dynamic obstacles.

In chapter 4, we compare the simulation result of the bacterial DGD model with experimental data. We computationally analyze the single cell tracks of the chemotactic bacteria cells and show that bacteria cells adapt their swimming and/or tumbling pattern in response to the environment they are in. Based on the observation from the experiment we further update our bacterial DGD model and show that the updated model can closely predict the bacteria chemotactic movement both at population and single cell level under different experimental setups.

In chapter 5, we introduce the shared memory consensus model for genome wide epigenetic modifications. We formally define the *consensus write-erase problem* in the context of histone modifications. We present an algorithm that is consistent with various biological assumptions and discuss its run time.

Lastly, in chapter 6 we summarize the contributions of the thesis and discuss challenges and future research direction in the field of distributed computing in nature.

Chapter 2

Distributed Gradient Descent in Bacterial Food Search

2.1 Introduction[1]

In this chapter we show that a variant of a commonly used machine learning coordination algorithm, *distributed gradient descent (DGD)*, can be used to explain how large bacterial population efficiently search for food. Similar to the regular gradient descent setting [46], [47], by sensing the food gradient, each cell has its own belief about the location of the food source (Figure 2.1a). However, given potential obstacles in the environment, as well as limits on the ability of each cell to accurately detect and move toward the food source in this environment, the individual trajectories may not produce the optimal path to the food source. Thus, in addition to using their own belief each cell also sends and receives messages from other cells (either by secreting specific signaling molecules or by physical interaction; Figure 2.1(b)), which it then integrates to update its own belief and determine subsequent movement direction and velocity. The process continues until all the cells converge to the food source.

While it is possible to observe that the group-based approach shortens the time it takes a

[1]This chapter is based on the paper [27]

single cell to reach the food source , and several aspects of the molecular pathways involved in the communication between bacterial cells have been studied [19], the *computations* that the cells perform have not been been well characterized. Current models [29],[26] are largely based on differential equation methods. While these can indeed lead to a fast, coordinated search, they do not fully take into account the dynamically changing topology of cells' interaction network over time. Furthermore, they make unrealistically strong assumptions about the abilities of cells to identify the source(s) of the messages and to utilize a large (effectively continuous valued) set of messages, which is unrealistic given the limited computational powers bacteria cells possess.

The proposed DGD model makes biologically realistic assumptions regarding the dynamics of the agents, the size of the set of the messages, and the agents' ability to identify senders. We show that our DGD model solves the bacterial food search problem more efficiently (in terms of the overall complexity of messages sent) and more quickly (in terms of the time it takes the cells to reach the food source) when compared to current differential equation models. We argue that our model is in fact a distributed pseudogradient descent method[2], and hence we can adapt proof by [48] to show that our model converges to a local minimum under reasonable assumptions on how bacteria communicate. Simulation studies indicate that the solution is feasible and leads to improvements over prior methods.

2.2 Methods

2.2.1 A computational framework for understanding bacterial food search

While bacterial food search has been extensively studied, at both molecular and cellular levels, most studies have focused on the individual cell, rather than characterizing the collective performance of a bacterial chemotactic cells. To formally analyze this process and to derive methods

[2] *pseudogradient* here refers to the fact that each agent's *expected* direction of movement at each time step is a descent direction, even though it may not be a descent direction in actuality due to stochasticity.

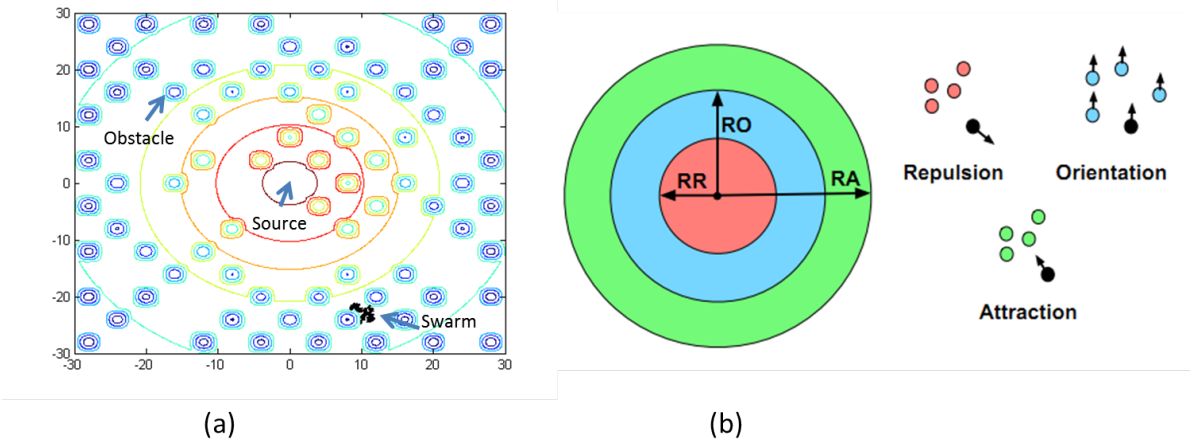

Figure 2.1: **(a):** Terrain model for bacterial food search. Obstacles are randomly placed and the food source is at the center of the region. Contours display the diffusion of the food source gradient. **(b):** Dynamics of repulsion, orientation, and attraction for a single bacterial cell in the Shklarsh et al. model. RR = radius of repulsion, RO = radius of orientation, and RA = radius of attraction.

that can be used for other tasks performed by severely restricted agents, we consider collective bacterial food search as a DGD algorithm for determining the direction of movement for each agent. DGD is a classic distributed optimization algorithm for finding a minimum of an objective function. In our case, the minimum corresponds to the food source, and the objective function incorporates the terrain over which the bacteria cells search. DGD is based on *message passing* between nodes (individual cells or agents) in a graph. These messages contain information that each node uses to update its own movement direction and velocity. The goal of DGD is for all nodes to converge to a single location in the search space.

While DGD has been studied in several different application areas, there are several differences between standard DGD algorithms [49, 50] and bacterial food search. First, classical algorithms assume that message passing occurs on a fixed and static network, whereas for bacteria edges and their weights are based on a sphere of influence which changes over time based on each bacterium's current location. Second, classical algorithms typically do not limit message

complexity (in terms of message size and number of messages sent), whereas bacteria have a limited message vocabulary [51] and may attempt to minimize the number of messages sent to conserve energy. Third, classical algorithms require significant data aggregation at individual nodes, whereas bacteria are not believed to collect and store data in this way. Considering that bacteria have been solving this optimization problem under these constraints for billions of years, we hope to learn interesting algorithmic strategies from studying this process. Such algorithms may address a new class of problems that may be important in other applications, such as swarm robotics [52].

For bacteria, the movement of an agent i at time n is a function $d_i(n)$ of two quantities: its own sense of direction $\theta_i(n)$ (based on the chemical gradient), and the locations and movement directions of other cells. Nodes in this graph correspond to bacteria, at some current location $x_i(n)$ and edges, representing physical distance, exist between two bacteria that lie within a sphere of influence of each other. The challenge lies in updating $d_i(n)$ based on the individual belief and the beliefs of neighbors, while using simple messaging (formalized below). For bacteria, messages passed along edges in the graph contains both homophilic components (attraction and orientation) and a heterophilic component (repulsion). Below, we first present prior work by Shklarsh et al. [26] that describes bacteria food search using a differential equation (DE) model and then present our DGD model, which improves upon the Shklarsh DE model in performance while relying on weaker, more biologically realistic, assumptions.

2.2.2 The Shklarsh model: Cell based differential equation computation

Initial models for bacterial chemotactic movements assume that cells solve a system of differential equations (DE) to determine their next move [26]. We briefly review this model below. The model assumes that individual cells follow a chemical gradient of food source by decreasing their (random) tumbling frequency at high concentrations and thus largely move in the direction of the food. Specifically, the bacteria perpetually moves in a direction which it repeatedly per-

turbs randomly. The frequency and magnitude of these perturbations are inversely related to the change in the food concentration between iterations, with the rough effect that the agent continues to move in directions along the gradient. Formally, under these assumptions, at time n, the bacterium changes its direction by an angle $\theta(n)$, which is a function of $\Delta c(n)$, the difference in food concentration between the current and previous time steps. Specifically, the new tumbling angle $\theta(n)$ is sampled randomly from a Gaussian distribution $\theta(n) \sim N(\theta(n-\Delta n), \sigma(\Delta c(n))^2)$ centered at the previous angle $\theta(n-\Delta n)$, with the variance $\sigma(\Delta c(n))^2$ given as:

$$\sigma(\Delta c(n))^2 = \begin{cases} 0; & \Delta c(n) \geq 0 \\ \pi; & \Delta c(n) < 0 \end{cases} . \tag{2.1}$$

Thus, based only on its own perception of the food gradient, the i^{th} agent updates its location $x_i(n)$ according to

$$x_i(n+\Delta n) = x_i(n) + s_i(n) \cdot \Delta n, \tag{2.2}$$

where $s_i(n)$ is the unit vector in the direction of the movement.

$$s_i(n) = (\cos(\theta(n)), \sin(\theta(n))). \tag{2.3}$$

So far we have discussed movements based on sensing by individual cells (agents). However, in almost all cases, cells move in collective manner. Communication among members improves the ability of individual cells to handle obstacles in the direction of the food source leading to faster and more efficient ways to reach the food. The communication among agents is divided into three components: (1) *repulsion* from very close agents to avoid collision ; (2) *orientation* to match the direction of neighboring cells, and (3) *attraction* to distant agents to keep the population unified (Figure 2.1(b)). The model of Shklarsh et al. assumes that cells align their trajectory with the direction of the other cells if they are close enough, while being attracted toward cells that are relatively far away.

Denote by $u_i(n)$ the vector agent i computes using the messages from the other cells (in this model, movements are of fixed length and so the only variable in each iteration is the direction).

Then, if any other agents j are within the physical interaction (repulsion) range RR as shown in Figure 2.1(b), the Shklarsh model sets:

$$u_i(n) = -\sum_{j \in B_{RR}} \frac{x_i(n) - x_j(n)}{\|x_i(n) - x_j(n)\|}. \tag{2.4}$$

Otherwise, the Shklarsh model sets:

$$u_i(n) = \sum_{j \in B_{RO}} s_j(n) + \sum_{j \in B_{RA}, j \notin B_{RO}} \frac{x_i(n) - x_j(n)}{\|x_i(n) - x_j(n)\|}, \tag{2.5}$$

where the first sum is over all agents j in the orientation range RO and the second sum is over all agents in the attraction range RA (but not in the range RO). Next, the agent combines the messages it received with its own observation, resulting in the following modification of equation (2.2):

$$x_i(n + \Delta n) = x_i(n) + d_i(n) \cdot \Delta n, \tag{2.6}$$

$d_i(n)$, is the direction of movement of agent i at time step n. $d_i(n)$ is a function of the agents own sense of direction $s_i(n)$ and aggregate sense of direction from other agents $u_i(n)$.

$$d_i(n) = \frac{u_i(n)}{|u_i(n)|} + w s_i(n) \tag{2.7}$$

where w is a scalar weighting factor. While for simulation purpose we use a fixed value for Δn, in practical scenario the $\Delta n \to \epsilon$, where the update essentially becomes a differential equation in continuous time. In the proposed model below, while we use less stricter assumptions than Shklarsh model, the same continuous time interpretation could be drawn.

2.2.3 A distributed gradient descent model

While the model presented by Shklarsh et al. [26] captures the basics of bacterial movements, it suffers from several problems which make it unlikely to be used by real cells and, as we show in Results, less efficient. First, the model assumes that cells can determine their exact distances from *each* other cell (as can be seen by the summations over neighbors cells in the orientation

and attraction ranges). This implicitly assumes that cells can *identify* individual senders, which is very unlikely given the large and dynamic nature of bacterial population. In addition, the model assumes that cells can interpret complex (real-valued) messages regarding the locations and orientation of other cells, which is also unrealistic [51]. Such assumption contradicts several studies that highlighted the limited computational power of bacteria cells. These studies indicate that bacteria computation is restricted to either signal summation or simple logical functions (AND or OR gates) [53, 54]. The way in which orientation signals are used in the Shklarsh model assumes that bacteria can perform complex computation which is unlikely given these prior results. Finally, the model assumes that, within each of the ranges above, each cell exerts the same influence regardless of their distance from the receiving cell, which is again unrealistic due to the nature of the communication channel (diffusion of a secreted protein). We have thus modeled bacterial food search using a DGD model that relaxes many of these assumptions while still allowing cells to (probably) reach an agreement regarding the direction of movement and eventually the location of the food source, as observed in nature.

Our model still distinguishes between physical interactions (leading to repulsion) and messages (secreted proteins). The former is handled by summing up the number of cells that are in physical proximity without relying on their exact location. However, we make several changes to Equation (2.5). First, we remove the requirement that cells identify the distance and direction to each other cell (and thus determine whether to use the attraction or orientation terms). Instead, we simply sum over all cells, taking into account their relative influences under the assumption that message strength decays exponentially with distance [55] (Fig. 2.2). Second, we discretize the messages that cells receive, resulting in simple messages with finitely many possible values. The changes lead to the following modification of equation (2.5):

Here, $D_{L,T}$ is a discrete thresholding operator parametrized by L, a positive integer denoting

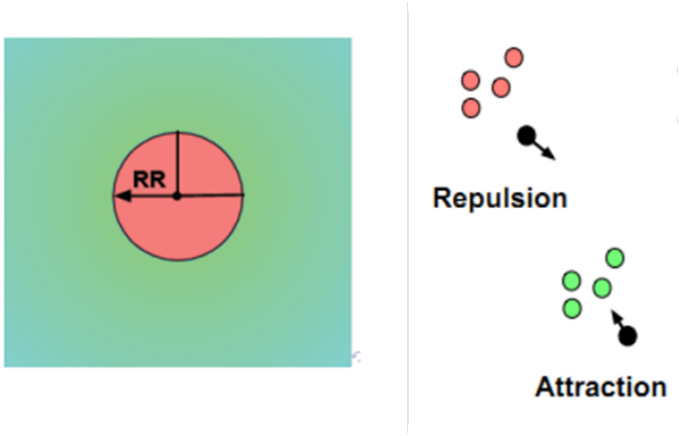

Figure 2.2: Dynamics of repulsion and attraction in DGD model. While we still maintain the physical (RR) versus communication (RO, RA) split between the repulsion and attraction / orientation information, our model does not assume that the identity of the sender is known and so it does not distinguish cells in the RO and RA locations.

the number of possible messages, and T, an upper bound above which all messages as treated as the highest value possible (see [56] for the exact construction of this "stone-age computing" threshold which has been used in ant models). C_a and C_o are positive diffusion constants, determining how quickly the attraction and orientation signals diffuse from the source agent. Typically, $C_o > C_a$, in which case nearby agents are influenced more by the orientation component and far away agents are influenced more by the attraction component.

Under this model, bacteria communicate attraction information using only $3 + \log_2 L$ bits to communicate ($\log_2 8$ bits for direction and $\log_2 L$ bits for magnitude). In addition, we also add a Gaussian noise component with a small variance σ to make this process stochastic. Note that the individual component of the agents' movement (based on the chemical gradient they perceive) is identical to that of the Shklarsh model, and that we are modifying only the communication model.

2.2.4 Proof of convergence

Unlike standard distributed gradient descent algorithms, the model described above does not rely on a fixed network. Instead, in each iteration, the topology of the network (and thus the weights placed on neighbors) changes with their movement. To prove that using such network (and the computation we assume) indeed leads to convergence as seen in nature (regardless of agents' starting locations), we adapt a convergence theorem for distributed pseudogradient descent from [48]. The convergence proof is only focused on the attraction component of the model (see Discussion). However, the theorem holds for both synchronous and asynchronous settings, and thus our model could be generalized to the case where agents' messages themselves travel stochastically.

The main idea of the proof of the convergence theorem is that there exists a sequence $\{y(n)\}_{n=1}^{\infty}$ such that, if, at time n, all agents were to cease to follow the gradient while continuing to move according to the attraction term (i.e., weighted averaging), then all agents would converge to location $y(n)$. For example, in the case that all agents communicate with equal weights, $y(n)$ is simply the mean of the agents' positions at time n. Under reasonable assumptions on the edge weights, information from each agent is likely to propagate throughout the population and so the agents will converge on $y(n)$ (plus individual noise), regardless of their starting position. Furthermore, once the agents are sufficiently close, the change in $y(n)$ (which is a weighted average of the gradients perceived by each agent) in each iteration becomes approximately proportional to $J(y(n))$, so that cells collectively behaves as a traditional gradient descent.

The proof relies on two lemmas. The first claims that agents continue to move in the correct descent direction once they have detected an increase in food gradient in one of their tumbles (until that direction ceases to be a descent direction). As we show, to achieve such expected decrease as required by the lemma for each step, we need to change either the tumbling distribution assumed in the original model, or the number of tumbles per step (i.e. prior to communicating a new location). While the former (a uniform tumbling distribution) is less likely in practice, the

latter (communicating only when a new descent direction is established) both makes biological sense [57] and, as we show empirically in Results, reduces the time it takes the cells to reach the food source. For the second lemma we again need to modify the original algorithm so that the step size (amount of progress made in the direction computed) is proportional to the detected gradient, causing the agents to slow as they approach the food source. This may be implemented in practice via a feedback loop used by bacterial cells [58].

After incorporating the variable step size we can write the complete update rule as following:

$$x_i(n+1) = x_i(n) + \gamma_i(n)(u_i(n) + s_i(n)) \tag{2.8}$$

where $s_i(n)$ is the belief of cell i at time n based on its sensing of the food gradient ($s_i(n)$ is a pseudogradient, as shown in Lemma 1 below). $\gamma_i(n)$ is the step size for agent i at time n, which, as discussed in the main text, is a function of the food gradient at the cell's location (a detailed formulation of $\gamma_i(n)$ is in the proof of Lemma 2 below).

In a simplified model that incorporates only the attraction term of communication, our convergence theorem follows from a general convergence theorem proven by Tsitsiklis et al. for a broad class of distributed pseudogradient descent algorithms (Theorem 3.1 and Corollary 3.1 of [48]). Work is needed only to show that our setting satisfies the six assumptions therein (listed and justified below). In order to match the notation of [48], we note that the update equation (using only the attraction term) can be written in the form

$$x_i(n+1) = \sum_{j}^{M} a_{ij}(n)x_j(n) + \gamma_i(n)s_i(n) \tag{2.9}$$

where $a_{ij}(n)$ is the edge weight between cells i and j at time n.

Convergence Theorem: Let $x_i(n) \in R^2$ denote the position of the i^{th} agent at time n and let $J : R^2 \to R$ denotes the concentration of the food source (the objective function). Under assumptions A1-A6 (see below), the following hold:

1. There exists a constant $\gamma^* > 0$ such that, if each step size is at most γ^* (i.e., $\sup_{i,n} \gamma_i(n) \leq$

γ^*), then, with probability 1, for all distinct agents i and j,

$$\lim_{n\to\infty} (x_i(n) - x_j(n)) = 0.$$

2. If, in addition, the level sets of J are compact, then, with probability 1, for all agents i,

$$\lim_{n\to\infty} \|\nabla_x J(x_i(n))\|_2 = 0.$$

3. Finally, if, in addition, all stationary points of J are minima (e.g., if J is convex), then, with probability 1, for all agents i,

$$\lim_{n\to\infty} J(x_i(n)) = \inf_x J(x).$$

The three components of the above theorem gives sufficient conditions for the agents to converge spatially, to converge to a connected set of stationary points, and to converge to a connected set of minima, respectively. Assumptions A1-A4 (stated precisely below) are that J is sufficiently smooth (its gradient is Lipschitz continuous) and that the agents' communication network is sufficiently dense so that information from any agent eventually propagates through the population. Assumptions A5 and A6 correspond to Lemmas 1 and 2 which we explicitly discuss below.

The lemmas, stated below, are needed to establish that our algorithm is indeed a distributed pseudogradient algorithm before we can apply the results of [48]. The lemmas essentially state that (a) the agents expect to move in a descent direction and (b) the variance in the agent's movements is at most proportional to the magnitude of the true gradient (so that, for example, it vanishes near stationary points and hence agents cease to explore once they have located an optima).

Assumptions

The assumptions needed for the general convergence proof are the following:

A1 There is some $\alpha > 0$ such that $a_{ij}(n) \geq \alpha$, for all times n and distinct agents i and j.

A2 There exists a constant B_1 such that the time between consecutive communications between any pair of distinct agents i and j is at most B_1 [59],[60].

A3 The number of messages communications between any two distinct agents during any duration of length B_1 is bounded by some constant B_2.

A4 J is continuously differentiable and its gradient $\nabla_x J$ is Lipschitz continuous.

A5 The conclusion of Lemma 1.

A6 The conclusion of Lemma 2.

Below we discuss why each of these holds in our setting.

A1 Since the edge weight $a_{ij}(n)$ is a positive, strictly decreasing function of the distance $\|x_i(n) - x_j(n)\|_2$ and the agents move only within a bounded region of R^2, such an $\alpha = \inf_{i \neq j,n} a_{ij}(n) > 0$ exists with high probability.

A2, A3 This point is somewhat subtle, and quite important for the generality of our work. In our model, each agent regularly (after a fixed number of rounds on individual movement, as discussed in the proof of Lemma 1 below) measures the *weighted average* of the signals from all other agents, through the update equation (3.6). The key observation here is that, in [48] (which only considers communication as being between pairs of agents, and hence phrases the assumption as above), each agent's update depends *only on the weighted average* of the messages received. Thus, this assumption is *weaker than* the assumption that the weighted average in the update equation incorporates *all* other agents (with weights satisfying the constraints of assumption A1). By construction, our model satisfies this. [3]

A4 Food is assumed to diffuse smoothly from its source (usually according to a Bessel, Gaussian, or exponential decay function of distance).

Lemma 1 Conditioned on the history of the algorithm, the expected gradient perceived directly by each agent (not accounting for information from the other agents) is a descent direction.

[3]Compare *Example V* of [48], which considers a similar communication pattern in a more abstract setting.

That is,

$$E\left[\frac{dJ}{dx}(x_i(n))s_i(n)\right] \leq 0.$$

In order to guarantee this we will allow agents to tumble a fixed number of times between rounds of communication, so that, in expectation, they will find and move along a descent direction.

Proof of Lemma 1: We first show the case when the direction after a tumble is chosen uniformly at random, in which case, the proof is straightforward. For sake of generality, we then show the case where the new angle is chosen according to any distribution (such a Gaussian) satisfying a certain weak uniformity condition, allowing multiple tumbles within each round of communication.

Uniform Case: In each iteration $n > 1$, if the previous direction $s_i(n-1)$ is not a descent direction (i.e., if $J(x_i(n)) \leq J(x_i(n-1))$), then $\frac{s_i(n)}{\|s_i(n)\|_2} \sim \text{Unif}(\partial B_1(0))$, [4] and the speed $\|s_i(n)\|_2$ is deterministic given $s_i(n-1)$ and $x_i(n-1)$. Since the function $s \mapsto \nabla_x J(x_i(n)) \cdot s$ is linear, its expectation over a uniformly distributed variable is 0, and so

$$[\nabla_x J(x_i(n)) \cdot s_i(n) | F_n] = \|s_i(n)\|_2 \left[\nabla_x J(x_i(n)) \cdot \frac{s_i(n)}{\|s_i(n)\|_2} \middle| F_n\right] = 0.$$

If the previous direction is a descent direction, the agent retains its previous direction.

General Case: Now consider a more general algorithm parameterized by a *tumbling distribution* with density D on $[-\pi, \pi]$ (above, D is uniform). In general, a single tumble may be insufficient to ensure that the expected resulting direction is a descent direction, and so multiple tumbles may be necessary. We assume that D is somewhat uniform in the following sense:

$$0 < c := \inf_{\theta \in [-\pi, \pi)} \int_{\theta}^{\theta + \pi/2} D(\phi_{2\pi})\, d\phi, \text{ [5]}$$

i.e., the agent has probability at least $c \in (0, 1/4]$ of tumbling toward any particular quadrant. This assumption certainly holds for the Gaussian tumbling distribution we use in the main paper, but also allows a broad range of other possibilities, including any distribution whose density is lower bounded away from 0 on $[-\pi, \pi]$.

[4] $B_1(0)$ denotes the unit ball centered at 0, and $\partial B_1(0)$ denotes it boundary.

[5] For $x \in R$, $x_\pi = ((x + \pi) \mod 2\pi) - \pi$ denotes the angle in $[-\pi, \pi]$ equivalent to x.

To prove the lemma, let $\theta_0 = \theta(\nabla_x J(x_i(n)), s_i)$, $\theta_{\ell+1} = \theta_\ell + \Delta\theta_\ell$, where each $\Delta\theta_\ell \sim D$ denotes the change in angle due to the ℓ^{th} tumble. [6] With probability in $(c, 1-3c)$, any particular $|\theta_\ell| \leq \frac{\pi}{4}$, in which case

$$\nabla_x J(x_i(n)) \cdot s_i(n) \geq \frac{1}{\sqrt{2}} \|J(x_i(n))\|_2 \|s_i(n)\|_2.$$

Also, with probability in $(2c, 1-2c)$, $|\theta_\ell| \geq \frac{\pi}{2}$ for all $\ell \in \{1, \ldots, k\}$, in which case, by the Cauchy-Schwarz inequality, $\nabla_x J(x_i(n)) \cdot s_i(n) \geq -\|J(x_i(n))\|_2 \|s_i(n)\|_2$. Otherwise, $|\theta_\ell| \in \left(\frac{\pi}{4}, \frac{\pi}{2}\right)$, so $\nabla_x J(x_i(n)) \cdot s_i(n) \geq 0$.

Let $L \in \{0, \ldots, k\}$ denote the last ℓ at which $\theta_\ell > \frac{\pi}{2}$ (i.e., after which, the agent maintains its current direction for the remaining $k - \ell$ tumbles). Since $L = \ell$ if and only if $|\theta_0, \ldots, \theta_\ell| \geq \frac{\pi}{2}$ θ_ℓ

$$\begin{aligned}
[\nabla_x J(x_i(n)) \cdot s_i(n) | F_n] &= \sum_{\ell=0}^{k} [\nabla_x J(x_i(n)) \cdot s_i(n) | F_n, L = \ell] [L = \ell] \\
&\geq \sum_{\ell=0}^{k} \|\nabla_x J(x_i(n))\|_2 \|s_i(n)\|_2 \left(\frac{1}{\sqrt{2}}(k-\ell) - \ell \right) 2^\ell c^{\ell+1} \\
&= \frac{c\|\nabla_x J(x_i(n))\|_2 \|s_i(n)\|_2}{\sqrt{2}} \left(\sum_{\ell=0}^{k} (2c)^\ell \left(k - \left(1 + \sqrt{2}\right) \ell \right) \right).
\end{aligned}$$

It is easy to see that this quantity is positive for sufficiently large k, since, as $k \to \infty$, $\sum_{\ell=0}^{k} \ell(2c)^\ell$ converges while $k \sum_{\ell=0}^{k} (2c)^\ell$ diverges (recalling $c \in (0, 1/4)$).

Lemma 2 The variance of the updates goes to zero as the cost function goes to zero. That is, for some $K_0 > 0$,

$$\left[\|s_i(n)\|^2\right] \leq -K_0 \left[\frac{dJ}{dx}(x_i(n)) s_i(n) \right].$$

Proof of Lemma 2: We assume here that the food follows an isotropic Gaussian distribution centered at the origin. That is, for some $\sigma > 0$, $\forall x \in R^2$,

$$J(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|x\|_2^2}{2\sigma^2} \right).$$

[6] For two vectors u and v, $\theta(u, v) = \cos\left(\frac{u \cdot v}{\|u\|_2 \|v\|_2}\right)$ denotes the (smallest) angle between u and v. For notational convenience, we measure angles as lying in $[-\pi, \pi]$, with 0 denoting the direction of $\nabla_x J(x_i(n))$.

Then, for all agents i and times n,

$$\frac{dJ}{dx}(x_i(n)) = -\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|x_i(n)\|_2^2}{2\sigma^2}\right) x_i(n)$$

Plugging in

$$x_i(n) = \sum_j a_{ij}(n)x_j(n) + \gamma_i(n)s_i(n)$$

gives

$$\begin{aligned}\frac{dJ}{dx}(x_i(n)) = &-\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|x_i(n)\|^2}{2\sigma^2}\right) \sum_j a_{ij}(n)x_j(n) \\ &- \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{\|x_i(n)\|^2}{2\sigma^2}\right) \gamma_i(n)s_i(n)\end{aligned}$$

Now the right-hand side of the lemma:

$$\begin{aligned}-K_0\left[\frac{dJ}{dx}(x_i(n))s_i(n)\right] &= K_0\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|x_i(n)\|^2}{2\sigma^2}\right) s_i(n) \sum_j a_{ij}(n)x_j(n)\right. \\ &\quad \left. + \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|x_i(n)\|^2}{2\sigma^2}\right) \gamma_i(n)s_i(n) \cdot s_i(n)\right] \\ &\geq K_0\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|x_i(n)\|^2}{2\sigma^2}\right) \gamma_i(n)\|s_i(n)\|^2\right] \\ &= \left[\exp\left(-\frac{\|x_i(n)\|^2}{2\sigma^2}\right) K'\gamma_i(n)\|s_i(n)\|^2\right] \geq \left[\|s_i(n)\|^2\right]\end{aligned}$$

when the step size $\gamma_i(n)$ satisfies $K'\gamma_i(n) \geq \exp\left(\frac{\|x_i(n)\|^2}{2\sigma^2}\right)$.

Bound on number of steps to converge

We next use the above analysis to provide a (loose) upper bound on the number of steps needed for convergence in the dynamic DGD model. Define by D_{max} the maximum distance of the source/minima from any point in the search space. Since the variable step size is constrained as $\gamma_i(n) \geq \frac{1}{K'} \exp\left(\frac{\|x_i(n)\|^2}{2\sigma^2}\right)$, a lower bound for the minimum step size would be when $\|x_i(n)\| \to 0$. Set that minimum step size as γ_{min}. Then, according to the constraints above, $\gamma_{min} \geq \frac{1}{K'}$.

Using these notations we can compute an upper bound on the number of steps (T_{conv}):

$$T_{conv} \leq \frac{D_{max}}{\gamma_{min}}$$

$$T_{conv} \leq \frac{D_{max}}{\frac{1}{K'}}$$

$$T_{conv} \leq K' D_{max}$$

2.2.5 Terrain modeling

We use a similar terrain model to the one used in [26]. Food density and terrain are stationary over time. Food is assumed to be diffused through the terrain, with a global maximum at the source. Specifically, we modeled the food density as an isotropic Gaussian function:

$$J(r) = \frac{K}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{r^2}{2\sigma^2}\right). \tag{2.10}$$

In addition, we introduce random obstacles in the form of local minima. In particular, we use the half cosine function to generate field of obstacles:

$$g(r) = min(0, -4(cos(\pi r/4) + 0.5),$$

from which we randomly remove a small number of obstacles to increase trial-to-trial variability.

2.3 Results

To determine whether the restricted communication model we assume can indeed lead to efficient convergence we performed several simulations of bacterial food search. First, we compare the search efficiency of bacterial food search with and without communication, and between our model and the Shklarsh et al. model. Second, we introduce multiple populations and test how their trajectories affect each other. Third, we explore the predictions of the model for a setting in which a fraction of the bacteria are behaving differently than the others. This latter point is of great current interest since 'cheaters' (cells that receive messages but do not spend the energy on sending them) may be responsible for a form of antibiotic resistance that has been recently observed [61]. In addition to that we also discuss the effect of various components of the

model on the bacterial food search performance, such as different communication components (repulsion, orientation, attraction), distance based weighting, and population size.

To save space, in the remainder of this section we refer to the Shklarsh et al. model as the 'DE' model and our model as the 'DGD' model (even though both rely on updating magnitude and direction in time steps).

2.3.1 Simulation parameters

The results presented in this thesis are produced with 30 agents (unless otherwise stated). For the simulation we use a Repulsion Radius of 0.1 and for the [26] model we use 4.0 for Orientation Radius and 4.3 for the Attraction Radius. The discretized thresholding operator $D_{L,T}$ in our model used $L = 4$ and $T = 3$ (see [56]).

All the probability distributions are generated from 300 independent trials. For each trial the number of iterations required for at least 90% of the agents to reach within a fixed radius of the food source was measured. We have set this source radius at 2.5. For modeling the food source we have used $\sigma^2 = 1000$ and $K = 200$ in equation (2.10).

Each population is generated by choosing *population center* at a certain fixed distance from the food source, with the angle along this distance circle chosen uniformly at random. Agents are placed uniformly in a small square centered at the population center, with the constraint that no agent starts on an obstacle. See Figure 2.1 for an example of a single trial initialization.

2.3.2 Performance on a realistic food search simulation

To evaluate the performance of our method we first tested it using the terrain and obstacle model stated above. In these settings we varied the number of agents, the communication between agents and the number of groups of cells.

The quantity we compared was the time it took cells to reach the food source (in terms of steps, since both algorithms can be run synchronously). Figure 2.3 presents the distribution of

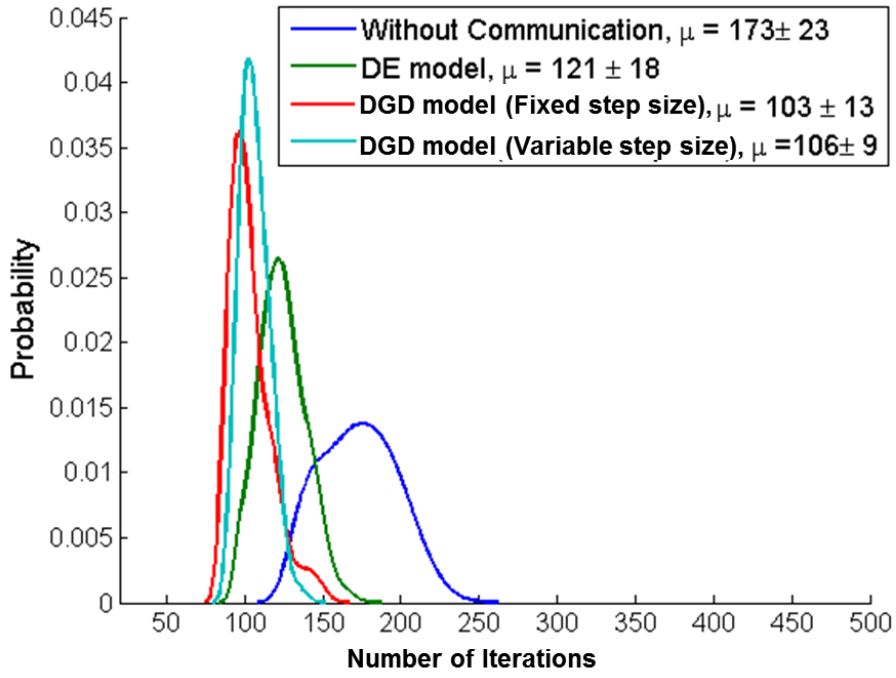

Figure 2.3: Comparison between the original Shklarsh model and the DGD model. μ denotes the mean number of iterations per agent, plus or minus the standard deviation.

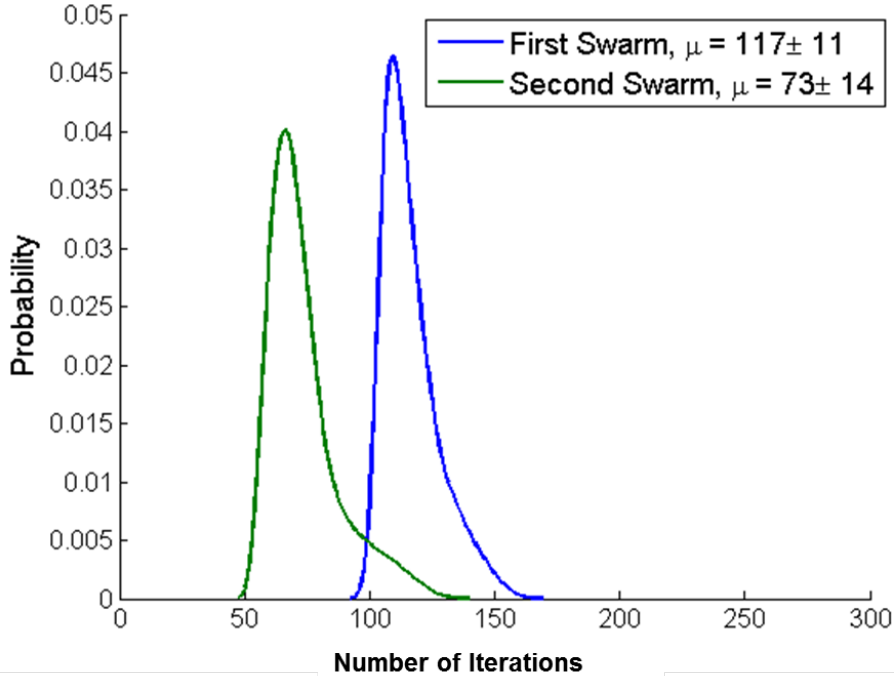

Figure 2.4: Distribution of number of iterations for two sequential groups using variable step sizes. μ denotes the mean number of iterations per agent.

the number of steps it takes cells to reach the food source under several different models.

The first is a model without communication (i.e. each cell can only sense the food gradient, but does not receive or send any message). The second is the Shklarsh et al. model with adaptive weighting, and the other two models are our DGD model with fixed or variable step size (the latter is required for the convergence proof above while the former is usually used in bacterial models). As can be seen, communication greatly improves the time it takes cells to reach the food source, which may explain why such a secretion-based communication system has evolved in this species. As for the specific communication model, the DGD model improves the results when compared to the DE model, even though our DGD model severely restricts the set of messages that can be used. In fact, the discretization of messages decreases both the mean and variance of the distribution. This is likely due to the fact that, by thresholding, the discretization step is effectively reducing the large noise that can be associated with individual messages. In addition, the distance-weighted edges (corresponding to the diffusion rates of secreted communication molecules) also improve the performance of the method. Simulation movies with and without communication between cells can be observed on the Supporting Website.

Figure 2.4 displays the effect on the time it takes cells to reach the food source if another group is added to the simulation. In this sequential set up, the second group starts 50 iterations after the first group. As can be seen, the fact that the first group was already able to successfully navigate to the food source enables the second group to utilize (at least partially) the trajectory they identified to further reduce the time it takes to reach the food. Interestingly, while the improvement for the second group is indeed large, we also see a *decrease* in the performance of the first group compared to the single population result presented in Figure 2.3. This is due to the negative influence that the second group has on the first when it enters the region of influence. Since the second group starts in the opposite direction of the food source, the first group is (partially) adjusting its direction incorrectly (based on attraction to cells in the second group) increasing the number of iterations it takes cells to reach the food source. We also tested a sce-

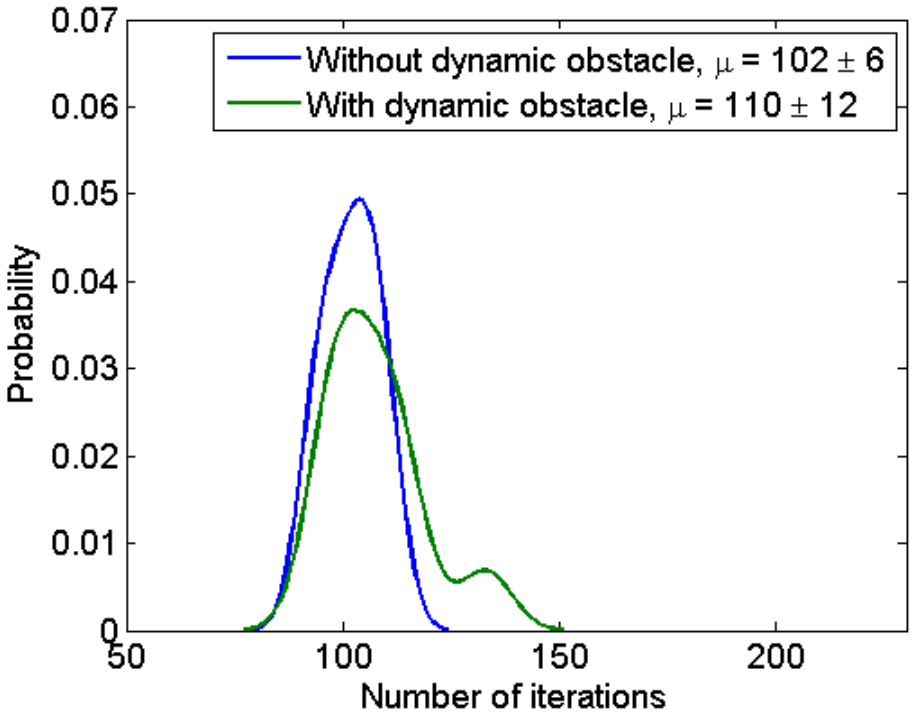

Figure 2.5: Performance of DGD with and without dynamic obstacles.μ denotes the mean number of iterations per agent, plus or minus the standard deviation.

nario where obstacles are added dynamically to the setup (Fig. 2.5). As can be seen from the figure the, addition of dynamic obstacle actually does not affect the performance much (number of iterations increases by 8% only). Since the algorithm does not assume that the location of obstacles are known a priori, the cells do not differentiate between existing obstacles and obstacles that are dynamically added. Hence the performance of the algorithm is mostly unchanged.

We also tested the performance of the algorithm with using a different weight function. So far we assumed that the weight of the messages decays exponentially with distance. Fig. 2.6 compares the performance of a simulation using such decay function to a simulation that uses inverse square decay ($weight \propto x^{-2}$). We have set the coefficient of decay C_a such that the average rate of decay is the same in both cases. As can be seen, the formulation of the weight function does not seem to affect the performance much.

2.3.3 Sensitivity to silent agents

Previous work has shown that some cells in a population become 'silent' [62],[63], [64]. These cells receive messages from the other cells but do not send messages themselves. While such behavior is beneficial from the individual standpoint (less energy is required to synthesize and secrete the signaling molecules) it may be harmful for the population as a whole since if these 'silent' cells proliferate the population will lose its ability to utilize communication to improve food search. Recent work has shown that stochastic activation of such silencing and other individual behavior mechanisms can explain how they can be advantageously used (for example, for developing antibiotics resistance) without affecting the overall ability of cellular coordination. We have thus used the obstacle model again to study the sensitivity of bacteria chemotaxis performance to the fraction of silent cells in the population. For this, we varied the fraction of silent agents from 0 to 1. As can be seen in Figure 2.7 up to a certain threshold we do not see a large impact for the increase in silent cells which supports the recent findings of [61]. Specifically, based on our models the performance of the population is only 20% less than optimal even

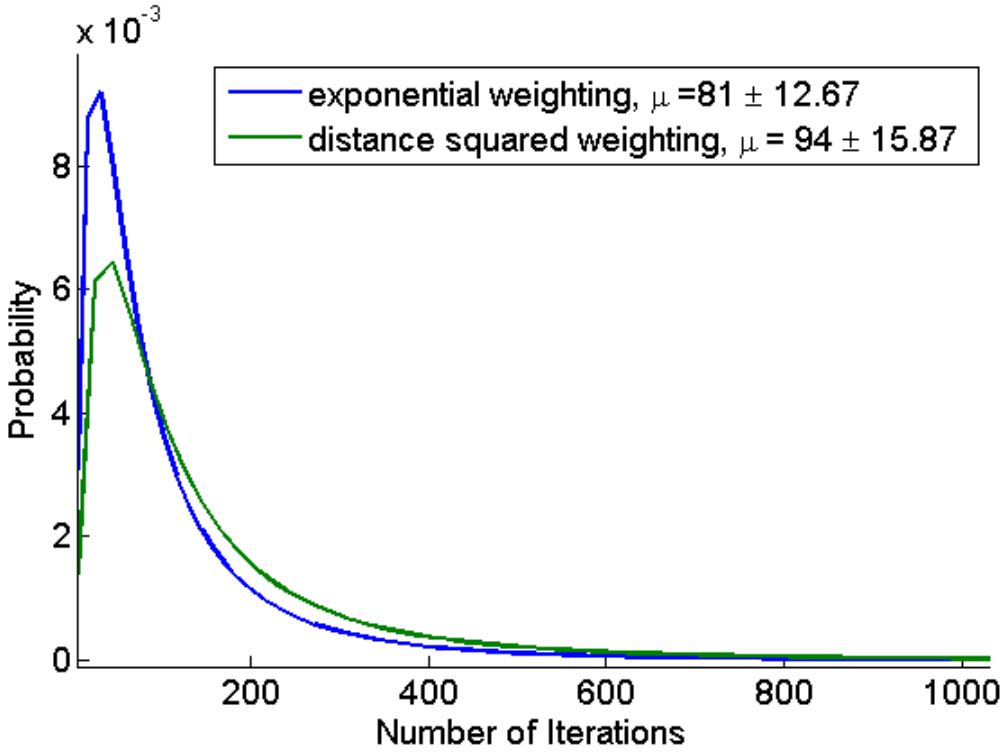

Figure 2.6: Performance of DGD with exponential weighting (proposed) and square law weighting.μ denotes the mean number of iterations per agent, plus or minus the standard deviation.

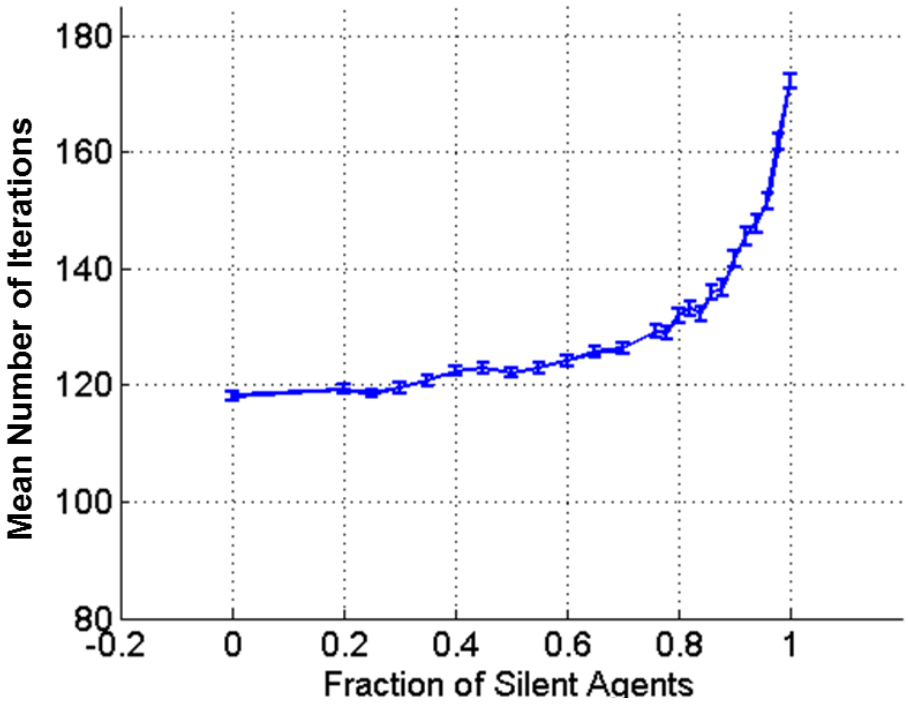

Figure 2.7: Mean path lengths over silent fractions, based on 100 trials. More silent bins were taken in the range of .7 to 1 to illustrate the phase transition. Error bars indicate standard deviations.

if 85% of the cells are silent. We also tested another case, where a fraction of the population sends wrong messages (see Fig. 2.8). However, unlike the silent population scenario, here the number of iterations to reach the food source continues to increase as we increase the fraction of wrong message sender. At high fraction ($> .5$), the search could not finish within the stipulated maximum iterations (300). Hence we can conclude that the bacterial search could be highly susceptible to wrong messages.

We have further analyzed various aspects of the communication model to determine the roles of each type of message being sent (orientation, attraction) or sensed (physical proximity), in terms of the impact on the time taken to reach the food source. See Supplement for details.

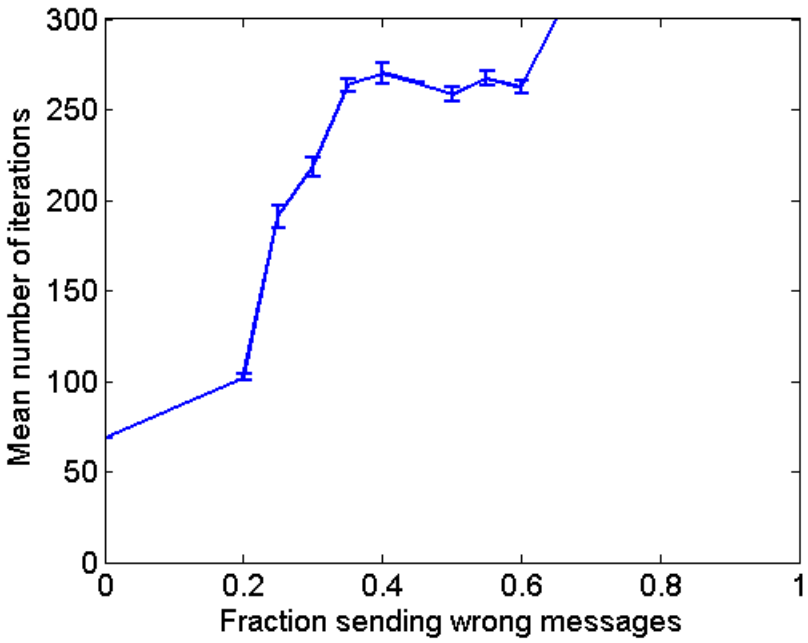

Figure 2.8: Mean path lengths over wrong message sender fractions, based on 100 trials. Error bars indicate standard deviations.

2.3.4 Effect of different modeling components

We have tested the effects of each of the signals the agent / cell utilizes as part of the DGD model. These include: i) repulsion information, ii) orientation, and iii) attraction. We have evaluated the behavior of the cells in the absence of either of these component to determine their impact of the ability of cells to efficiently reach the food source.

Figure 2.9 presents the simulation results for this analysis. As can be seen, while performance decreases when the orientation and repulsion components are disabled (average number of iterations required to reach the food source increases from 105 to 118 for repulsion and 105 to 115 for orientation), the effect is not large. In contrast the attraction component has a large effect on performance. Specifically, removing this component increases the mean path length (168 iterations) by 60%. One reason for this observation is that the decay in the weights of attraction components is one fourth of that of orientation (see methods section), providing opportunity for

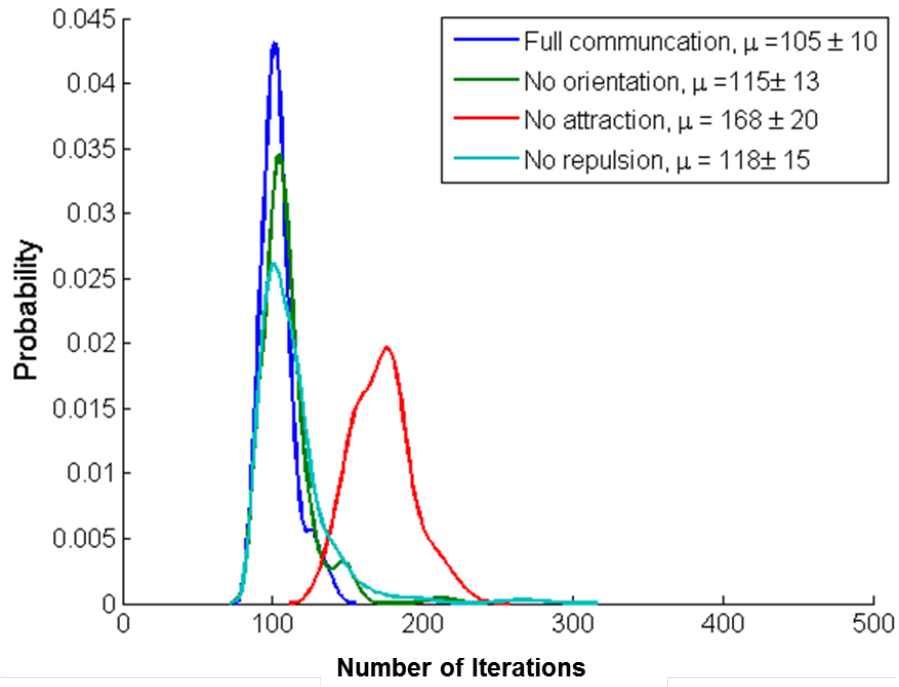

Figure 2.9: Effect of the removal of different communication components. μ denotes the average number of iterations required for each agent to reach the objective.

the cells to have a longer range of interactions which dominates their movement.

To test the impact of taking the distance into account when computing the local descent function we have run our method with and without using distance based weights, keeping all the other parameter settings same. Figure 2.10 presents the results of this analysis using two sequential groups. As can be seen, for both the first and second group, weighted communication leads to better performance than unweighted communication. Specifically, without using weights we see 10% and 22% increase in the number of iterations cells need to reach the food source for the first and second group, respectively. Thus, using the weighted version, which is also likely biologically correct, improves performance of such food search.

Another aspect we tested is the impact of population size. As can be seen in Figure 2.11), overall performance improves with the increase in population size indicating that communication helps cells reach their goal faster. However, depending on how we model the repulsion distance,

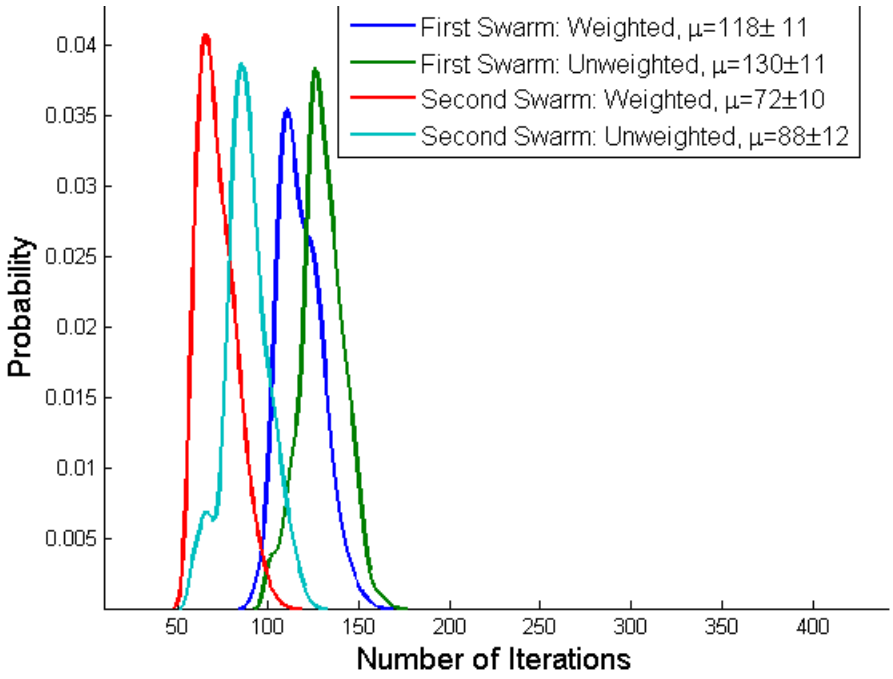

Figure 2.10: Comparison of bacteria chemotaxis performance using weighted and unweighted communication. μ denotes the average number of iterations required for each agent to reach the objective.

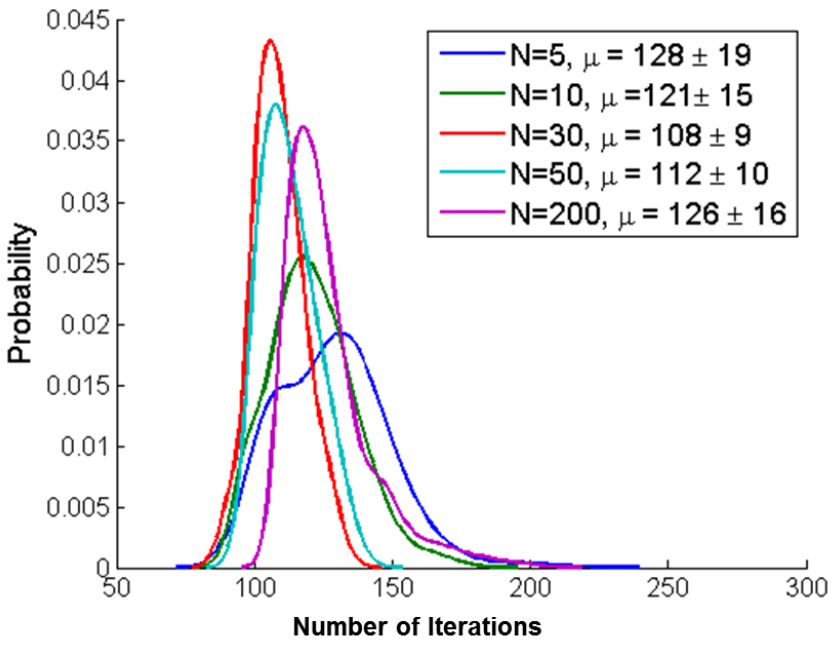

Figure 2.11: Distribution of path length over different population sizes. N denotes the number of agents in the population and μ denotes the average number of iterations required for each agent to reach the objective.

at some point such increase can lead to crowding. When the agents are too close to each other, their movement is highly constrained and mostly dominated by the repulsion effect. Therefore it takes much longer for a larger population to reach the food source. From our simulations we find that population size 30 to 50 gives the optimal performance.

2.4 Discussion

We have shown, both theoretically and in simulations and experiments, that a distributed gradient descent model can efficiently interpret the communication of agents under severe limitations. These include limits on the complexity of messages and the ability to identify which agent is sending the message, while at the same time assuming a dynamic environment where neighbors' locations (and their influence) change constantly. We have proved that, under reasonable

biological assumptions, the communication algorithm discussed is likely to converge, helping to explain how bacteria can efficiently coordinate food search in harsh environments and and improving upon prior models of this process. Moreover, the social dynamics of bacteria can also affect their ability to resist antimicrobial therapy, and inhibition of bacterial cooperation is an alternative approach to minimize collective resistance [65].

Our convergence proof only holds for the attraction information and does not hold for the repulsion and direction components of the computation performed by each agent. Since the orientation information is also a vector averaged over all neighbors, we believe that the proof can be extended to include this communication term as well, though a key challenge would be to understand the sometimes competing goals of local versus global improvement near certain obstacles. Repulsion is more difficult to analyze because it is not based on averages. However, as we show in the Results, the attraction term has by far the most significant effect on bacteria chemotaxis performance while the other two communication terms have a much less significant impact on the time it takes cells to reach the food source. Hence, for computational applications of our method, it may suffice to use the attraction term, in which case convergence is guaranteed. For subsequent application of the model in later chapters, we drop the orientation term since it affects the model performance the least.

In bacterial chemotaxis, the search process is primarily guided through the food gradient. While in this project relatively small sized terrain is considered (see more in Chapter 4), in other search tasks such as animals finding source of odor existence of the search space could be quite large. Often times because of winds and other environmental noise the gradient difference could be undetectable by the agent in different regions of the terrain, i.e., the gradient becomes noisy and patchy. In such cases estimating movement direction based on chemotaxis could be unreliable. Other recently proposed approaches such as Infotaxis [66] could be an alternative to that. Instead of looking at the gradient gain, it focuses on maximizing the information gain. While infotaxis is not a direct fit for our bacterial food search problem, it could be useful in other

search tasks where a reliable gradient difference detection is not guaranteed.

Chapter 3

A Bacterial based Distributed Gradient Descent Model for Mass Scale Evacuations

3.1 Introduction [1]

A potential application for the bacterial DGD model is designing mass scale evacuation strategy. Mass evacuation under emergency conditions warrants for a dynamic coordination method that requires very limited resources and relies on minimal assumptions. In such cases (for example, following fire or explosion in a crowded location) individuals have very limited ability to communicate and the infrastructure maybe damaged leading to severe constraints on the ability to send, receive and process information. To that end, in this chapter we present an extension of the Bacterial DGD model.

The extended model maintains all of the positive aspects of the bacterial model including the reliance on simple messages, data aggregation and the ability to handle noisy environments, for use in mass evacuation situations. Since the communication and movement paradigm is significantly different between human and bacteria, we have extended the DGD model to accommodate additional aspects of human movement, including variable speed (as opposed to the constant

[1] This chapter is based on the paper [67]

speed assumed for bacteria). In bacteria messages are communicated through secretion of proteins (via a chemical gradient). Here messages can be communicated through visual perception or through a simple bluetooth app capable of sending and receiving information about speed and location of agents. We analyze the new model and show that it is guaranteed to converge to a local minimum. We have tested the extended Bacterial DGD model in a number of simple and realistic simulations and compared its performance with the performance of prior methods suggested for the same task including the original social force model [32] which does not use communication between the agents and the more recently proposed modifications including the modified social force model [38], [33] and the single neighbor interaction model [37], [36]. In the no communication social force model, agents move towards the closest exit in a shortest possible path. Their movement however is influenced by repulsive force from the neighboring agents and walls. The repulsive force only plays a role when the agent is inside the repulsive zone of the neighboring agent. In the modified social force model, this strict repulsive zone is relaxed. In this model the repulsive force gradually decays as the agent moves further from neighboring agent. However, both of these models do not assume or use any other form of messages other than repulsive message. The single neighbor algorithm selects the closest neighbor outside of its repulsive zone. Then it combines the location and speed information of the selected neighbor with its own speed and location to determine the next move. As we show, in dynamic complex environments the DGD model achieves the best performance when compared to prior methods while still only using simple messages and computation, making it a viable alternative for such situations. To mimic the mass evacuation scenario, we have ran the simulations with 300 agents which is in line with the prior experimental studies [68], [69] on mass evacuation. The experiments were conducted on buildings of different structures and consisted of 177 [68] to 294 [69] participants.

3.2 Method

3.2.1 Computational model

We extend the bacterial based DGD model for performing efficient emergency mass evacuations. The original Bacterial DGD model assumes that individual cells can sense the chemical gradient of a food source and decrease their (random) tumbling frequency at high concentrations allowing them to move in the direction of the food. The frequency and magnitude of these perturbations are inversely related to the change in the food concentration between iterations, with the rough effect that the agent continues to move in directions along the gradient. In case of mass evacuation, we assume that such individual movement direction can be computed from the prior knowledge of environment (such as route to the nearest exit). In addition to the individual movement, communication among cells is another component in the Bacterial DGD model. Communication is achieved by cells that secrete protein molecules that are detected by neighboring cells. Through these molecules the cells can send information about their current location. Cells combine their own observations of the food gradient with the information they receive from other cells to determine the direction of their next movement. The ability to integrate information from neighboring cells helps each cell to overcome issues related to noise and obstacles that can block direct access to the food source and usually lead to faster convergence [26]. Note that, while several cells secrete these proteins, the receiving cell cannot differentiate between the senders (all cells secrete the same type of molecule) and so each receiving cell only observes the aggregate of all messages sent. We use similar simplistic message aggregation in the extended DGD model as well. However there are some key differences in the setup of the extended Bacterial DGD model for human evacuation.

Our extended DGD model relies on the following assumptions: i) each agent has access to a static map of the environment it is in (though the agent is not necessarily aware of dynamic changes that occur, for example a fire at a specific place); ii) in addition to location information

each agent can send simple messages to nearby agents about their speed as well; iii) each agent can adjust their speed based on how far they are from their intended exit (in contrast, in Bacterial DGD the cells use a constant speed).

We assume a synchronous execution model where computation proceeds in rounds. During each round the agents decide on their next move (direction and speed) which is executed at the beginning of the next round. At each round computation is based on the current agent location and on messages received during that round.

3.2.2 Individual environment map and gradient

We model the environment as a Gaussian gradient diffusing through the topology (see Figure 3.1). The minima of the gradient is at the exit (or at all exits if multiple ones exist) and it decreases as the agents gets further from it. The direction of movement of agent i at time n is denoted by $s_i(n)$. This direction can be computed individually based on the prior knowledge about the environment, for example by having the agent move in the direction of the nearest exit. However, since the nearest exit may be blocked this is not necessarily the optimal move for the agent. Following assumption 3 (each agent can adjust their speed based on how far they are from their intended exit), the speed of the agent is determined by its distance from the nearest exit. The further the agent is from the exit the faster it will move towards it. While such requirement is related to the ability to prove convergence, it is also useful in practice. The closer the agent is to the exit the more populated the area is and so lower speeds would avoid collisions that are likely if all agents move at a fast speed (see Results). Hence, when using only the agents knowledge it will update its location according to the following equation:

$$x_i(n+1) = x_i(n) + \gamma_i(n)s_i(n). \tag{3.1}$$

Here $\gamma_i(n)$ is the step size of agent i and $\gamma_i(n) = \exp(-k_\gamma g_i(n))$. $g_i(n)$ is the geodesic distance of agent i from the nearest exit. Following assumption 1 (each agent has access to a static map of the environment it is in), this can be computed by all agents ahead of time from their known static

environment map. We are using geodesic distance metric instead of regular Euclidean distance metric to account for the topology of the environment. Discussion on geodesic distance metric and performance comparison of the proposed method using the Euclidean and geodesic distance metric are presented in the Results section. The adjustable speed $\gamma_i(n)$ is formulated following assumption 3. k_γ is the coefficient of the exponential weighting. This hyperparameter should be set as a function of the environment. $x_i(n)$ is the location of agent i at time n.

3.2.3 Communication between agents

According to the second assumption of our extended DGD model the agents can communicate with each other. The messages have two components, i) location, and ii) speed. The agents adjusts both its direction and speed based on messages from nearby agents. The speed information is one of the key differences between the bacteria DGD model (which does not use speed) and the human evacuation model described here. In the extended model we allow agents to change the speed according to the distance from the exit and the speed of neighboring agents. Variable speed allows agents to move faster in less congested areas (farther from the exits) and to slow down when reaching likely congested areas (near the exits or when others in their neighborhood are moving slower). This can improve evacuation times as we show in Results while still leading to (guaranteed) convergence.

The advantage of communication in this setting is that other agents may be aware of additional information that the agent itself does not have (for example, they have observed a fire or an explosion at a specific place and are moving away to another exit). As noted above agents cannot directly distinguish between other agents and messages are aggregated either at the receiving end for each agent (removing the need for agents to know who their neighbors are) or using intermediate devices such as sensors located at various points in the room / area. Thus, no assumptions are needed about setting up groups or knowledge of the identity and number of other agents which greatly simplifies the algorithm [70], [71]. This is accomplished by updating the direction

with aggregated (summed) location information from others. Messages are weighted by the distance between the agents, such that closer agents will have more impact compared to the agents that are further (for example by using approximate location of senders w.r.t to receiver). Using these messages agents update their individual direction, $d_i(n)$ as following:

$$d_i(n) = \sum_j \exp(-C_a|x_j(n) - x_i(n)|)(x_j(n) - x_i(n)) + s_i(n). \tag{3.2}$$

Note that this $d_i(n)$ is different from $s_i(n)$. $s_i(n)$ is the direction obtained solely from individual sensing of the gradient, whereas $d_i(n)$ is the aggregate direction after integrating the agent perceived gradient with messages from other agents. Similarly each agent also updates the individual step size $\beta_i(n)$ by combining messages from other agents (following assumption 2):

$$\beta_i(n) = \sum_j exp(-C_b|x_j(n) - x_i(n)|)\gamma_j(n). \tag{3.3}$$

Here the sum is over all agents the receiver has heard from and C_a and C_b are the coefficient of the exponential weighting for location and speed. The values of these hyperparameters should be set as a function of the specific environment the application is to be used in since they relate to the impact that distant agents can have on each other.

Both d_i and β_i (combined messages from others for location and speed, respectively) are discretized for low communication and computation overhead.

$$d_i(n) = D_{L,T}\left(\sum_j \exp(C_a|x_j(n) - x_i(n)|)(x_j(n) - x_i(n)) + s_i(n)\right), \tag{3.4}$$

$$\beta_i(n) = D_{L,T}\left(\sum_j exp(C_b|x_j(n) - x_i(n)|)\gamma_j(n)\right). \tag{3.5}$$

$D_{L,T}$ is a discrete thresholding operator, parameterized by L, the number of possible messages, and T, an upper bound above which all messages as treated as the highest value possible (see [56] for the exact construction of this stone-age computing threshold which has been used in ant models). Under this model, agents communicate location and speed information using only

$3+log_2 L$ bits. Using the messages the complete update equation for the new position of an agent is:

$$x_i(n+1) = \sum_j a_{ij}(n)x_j(n) + (\sum_j b_{ij}\gamma_j(n))s_i(n), \tag{3.6}$$

where $a_{ij}(n)$ and $b_{ij}(n)$ is the exponential edge weight between cells i and j at time n. $a_{ij} = exp(C_a|x_j(n) - x_i(n)|)$ and $b_{ij} = exp(C_b|x_j(n) - x_i(n)|)$.

Finally, the model includes a physical repulsion term so that agents do not bump into one another. If an agent physically touches another agent it moves in the opposite direction to where it came from regardless of the other information it has. Hence the direction will be updated as following:

$$d_i(n) = -\sum_{j \in RR} (x_j(n) - x_i(n)). \tag{3.7}$$

Here RR is an influence radius that is related to the physical size of the agents and their speed such that agents cannot reach agents outside of RR in a single round. In all other cases (when the other agent is not within the RR) the direction d_i is updated according to eq. 4. The dynamics of repulsion and attraction is shown in Fig. 2.2 in chapter 2.

3.3 Experimental results

While the convergence proof discussed in the previous chapter guarantees that agents eventually reach a global optimum, it does not provide guarantee on the time it would take them. Since time to convergence is a critical issue for many applications, we tested the usefulness of a bacterial based DGD method for mass scale evacuations in terms of the evacuation time. For this we considered two environments, a basic rectangular room with two exits (Fig. 3.2a) and the real floor plan of a building (Fig 3.3a). In both cases we have considered additional obstacle in the environment that are unknown a-priori to the agents. In each of these simulations we perform 10 independent trials starting with 300 agents placed uniformly at random in the room / floor (Fig.

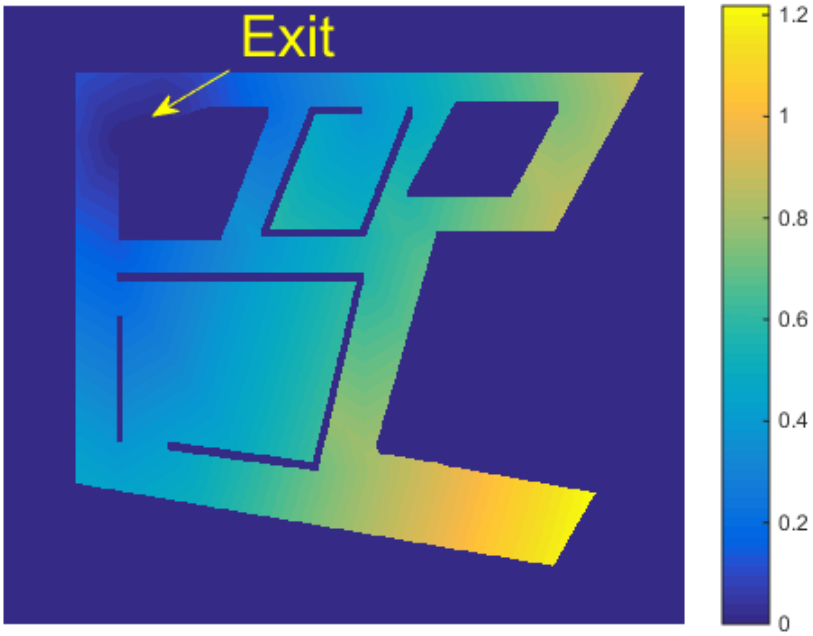

Figure 3.1: Gaussian cost function of a real floor plan with one exit. The minima of the cost function is located at the exit. The cost function increases as the geodesic distance from the exit increases.

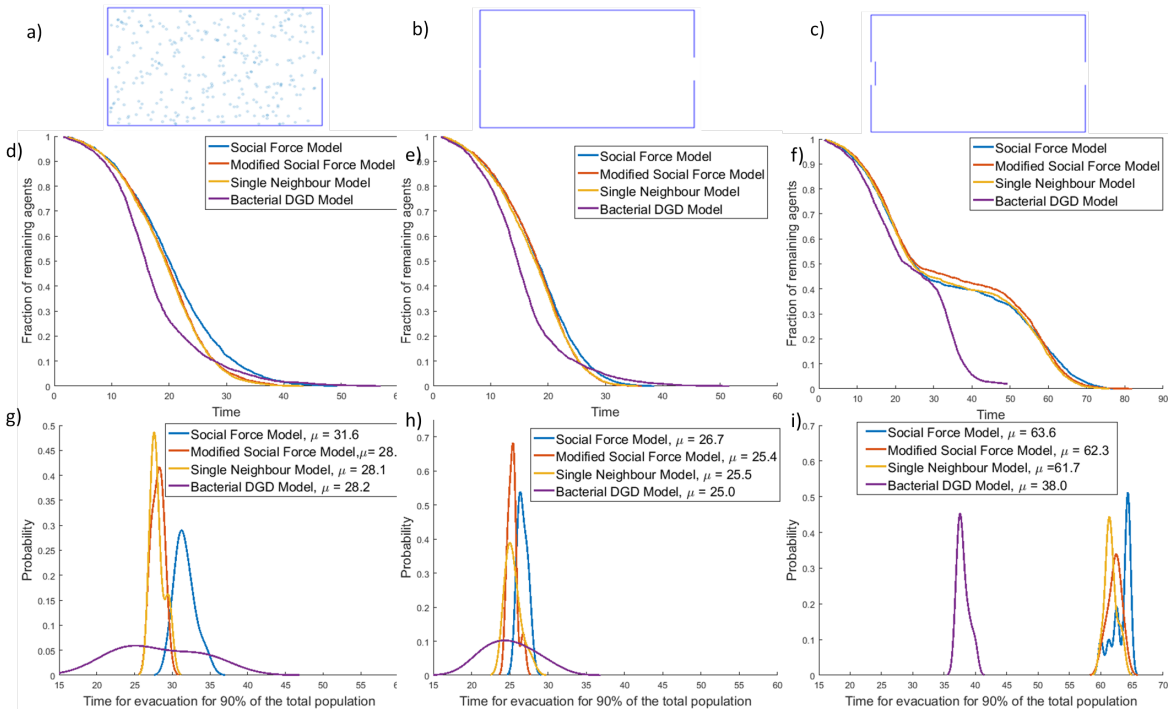

Figure 3.2: Simulations of a basic rectangular room. a) Basic room with equal sized doors, agents are randomly distributed initially. b) Room with one partially blocked door, and c) Room with one completely blocked door. d-e-f) Plot of average fraction (over 10 independent trials) of remaining agents at any time for regualr setup, partially blocked door setup, and completely blocked door setup, respectively. The Y axes shows the percentage of agents that are still in the room and the X axes plots the time in terms of iterations. g-h-i) Distribution of required time to evacuate 90% of the total agents in the regular setup, partially blocked door setup, and completely blocked door setup. In all of the plots the performance of the Bacterial DGD model is shown in comparison to prior models, such as social force, modified social force, and single neighbor model. The Bacterial DGD model has similar evacuation time to the competing methods in simple setup with no obstacle. In the presence of obstacles the Bacterial DGD model convincingly outperforms the competing models.

3.2a). Performance is evaluated by plotting the average fraction of agents that are still inside the room / floor at any time (thus, lower values represent higher evacuation rates). We have also plotted the distribution of the required time for the evacuation of 90% of the total agents in the different runs. In terms of comparisons, for each setup we compared the performance of the extended Bacterial DGD method with both, methods that model human behavior and prior algorithms proposed for general mass scale evacuations. These include the *social force model* [32] with no communication, *modified social force model* [38], [33] with distance weighted repulsive force, and the *single neighbor algorithm* [37], [36] in which each agent selects a single neighbor and uses the information from that neighbor to adjust its own belief.

3.3.1 Basic room with varying door size and obstacles

As a baseline we first tested all methods in simulations of a simple rectangular room with two doors of equal size on opposite sides (Fig. 3.2a). Since we do not include any unforeseen obstacle in this setup (i.e. the initial static map of each agent is the same as the actual room setup) we do not observe a large difference in average evacuation time between methods that use and do not use communication. However, while the average is comparable between methods, the variance is larger for the DGD method (lower panel of Fig. 3.3) with some runs taking longer than other methods we compared to. This highlights the tradeoffs of the DGD model. On the one hand it performs very well in complicated settings when other methods fail (see below). But on the other it can lead to slight increase in total time in simple settings due to the overhead it incurs (time to reach a consensus on the correct direction).

Next, we introduced dynamic changes to the environment (which are unknown to the agents until they reach them). In the simpler case (one door is partially blocked, middle column) the average time to evacuate 90% of agents is again largely comparable between the three methods that allow at least partial communication (the original social force model takes slightly longer) (Fig 3.2b,e,h). In contrast, when we introduced obstacles we saw large differences between the

performance of the DGD model and all other methods. In this setup we added an obstacle that completely blocked one of the doors (Fig 3.2c). Here we see that the average time to evacuate using the DGD model is almost 40% faster than all other methods (38 simulation rounds vs. 62 rounds for the other methods, Fig 3.2f-i). Since the DGD model utilizes information from neighboring agents, the agents can efficiently anticipate the obstacle even if they do not directly see it. Based on the messages they receive they can change their intended exit early leading to a much more efficient evacuation performance both in terms of evacuation rate and evacuation time. In the other models, the agents only changed their intended exit after individually observing the obstacle. This creates significant crowding at the blocked exit and subsequently increases the overall evacuation time.

3.3.2 Real floor plan of a building

We have next tested the method on a more realistic setup. For this, we constructed a map of the eighth floor of our departmental building denoting rooms, corridors and possible exits (see supplement for full floor plan). We evaluated both static environment (no obstacles, Fig. 3.3a) and dynamic changes where one of the corridors is blocked near one of the exits (for example, due to fire, Fig. 3.3b). The floor has three exits (marked by green). We assume that the agents know the floor topology and the route to the exits from any location in the floor. Unless faced with an obstacle (marked by red), the agents move towards the exit with the shortest geodesic distance. Again, when no obstacles are present we see that the other models that do not fully rely on communication lead to a slightly faster evacuation times when compared to the DGD method (Fig. 3.3c-e). As mentioned above this is likely due to the 'overhead' that the DGD methods introduces. Specifically, the slowdown is caused by an increase in bottlenecks resulting from the attraction term between the agents. The bottleneck is more pronounced here than the basic room setup because of the longer and more complex path towards the exits. This additional overhead is common to other models that use communication between agents. However, in cases where

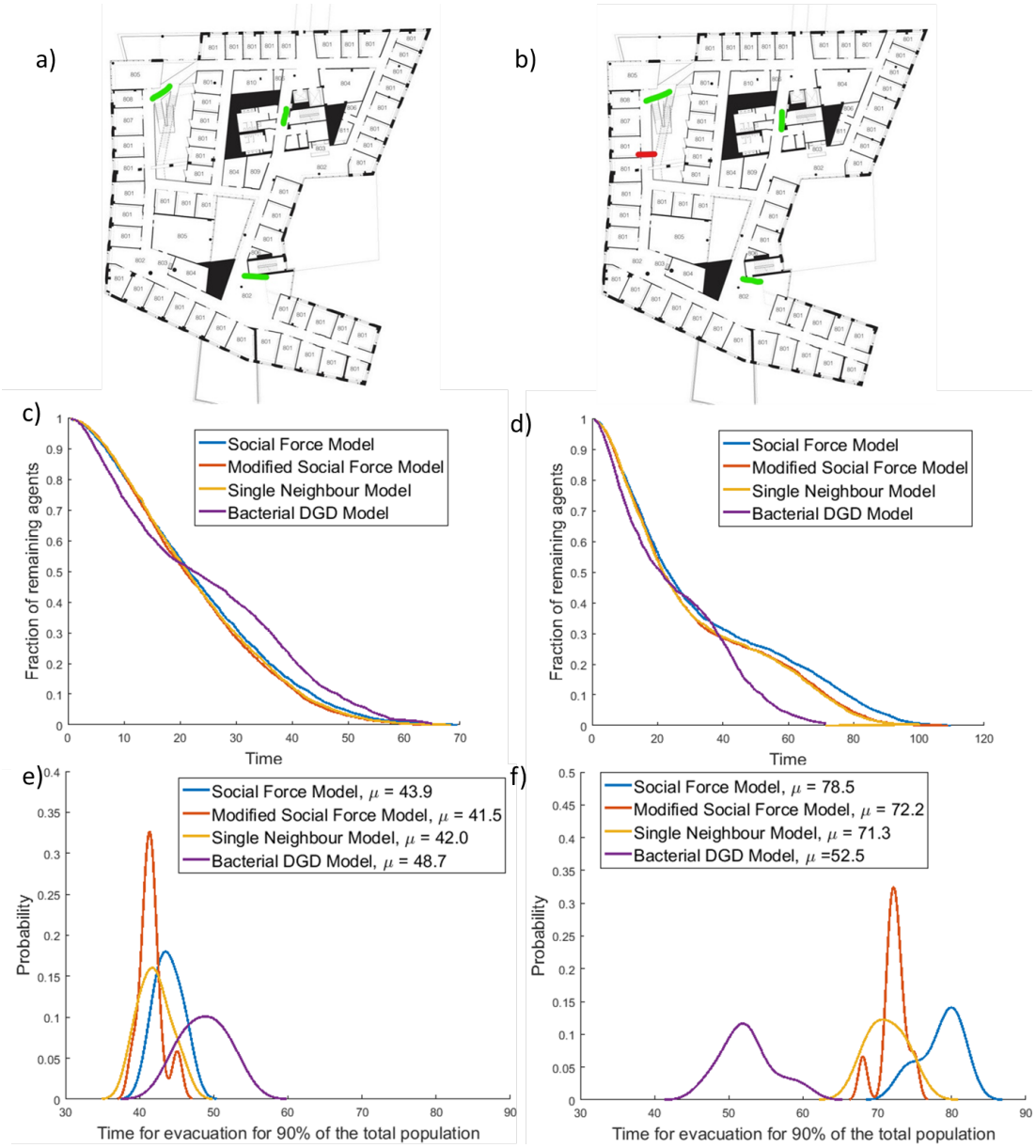

Figure 3.3: Simulation with the real floor plan of the 8^{th} floor of our departmental building. a) Floor plan. The green marks show the three exits of the floor. b) Floor plan with added obstacle. The red mark shows the location of the obstacle blocking one of the corridors to the exit. c-d) Plot of average fraction (over 10 independent trials) of remaining agents at any time for regular floor plan and floor plan with obstacle, respectively. e-f) Distribution of required time to evacuate 90% of the total agents in the regular floor plan and floor plan with obstacle, respectively. In all of the plots the performance of the Bacterial DGD model is shown in comparison to prior models, including social force, modified social force, and single neighbor model.

obstacles or other constraints are dynamically added, as is the case in the setting in Fig 3.3b, we observe a very large improvement when using the bacteria based DGD model. We see that the average overall evacuation time is only 52 simulation rounds for the DGD model (only 10% longer than when no obstacles are present). In contrast, evacuation time when using the social force model is 50% longer (78 simulation rounds) and the time for the modified social force and single neighbor model is also much higher (71$\sim$72, a 40% increase).

To further corroborate the importance of communication we have also compared Bacterial DGD model with individual sensing only (Fig. 3.4). The individual sensing model uses only eq . 1 to update the agent locations. It does not use any communication or repulsion between the agents. As can be seen from the figure, the individual sensing performs significantly worse than the Bacterial DGD model. Hence we can conclude that communication plays a crucial role to efficiently evacuate people in dynamic scenarios.

As we can see from the Fig. 3.3, there is an additional overhead when communication is used between the agents. We tested whether same effect is observed in other models using communication. Fig. 3.5 shows the comparison of performance of Bacterial DGD with another model that also require communication between the agents. The key difference between the compared and the proposed DGD model is the distance based weighting. In Bacterial DGD the weight of messages between the agents decreases with increasing distance between them. In the compared full communication model we are using a simpler version of communication without weighting the messages, i.e., the messages from all the agents have equal weight. As can be seen from Fig. 3.5 the unweighted communication model has a long tail distribution as a result of the incurred communication overhead (longer than Bacterial DGD). In addition to that, it has higher evacuation time in both no obstacle and with obstacle scenario.

In most of the prior models of evacuations, repulsion is used as a key component of the model [32], [38], [33]. Our evacuation model is inspired from bacteria chemotaxis movement, where repulsion is a key governing mechanism of the movement [26]. We have kept the repulsion

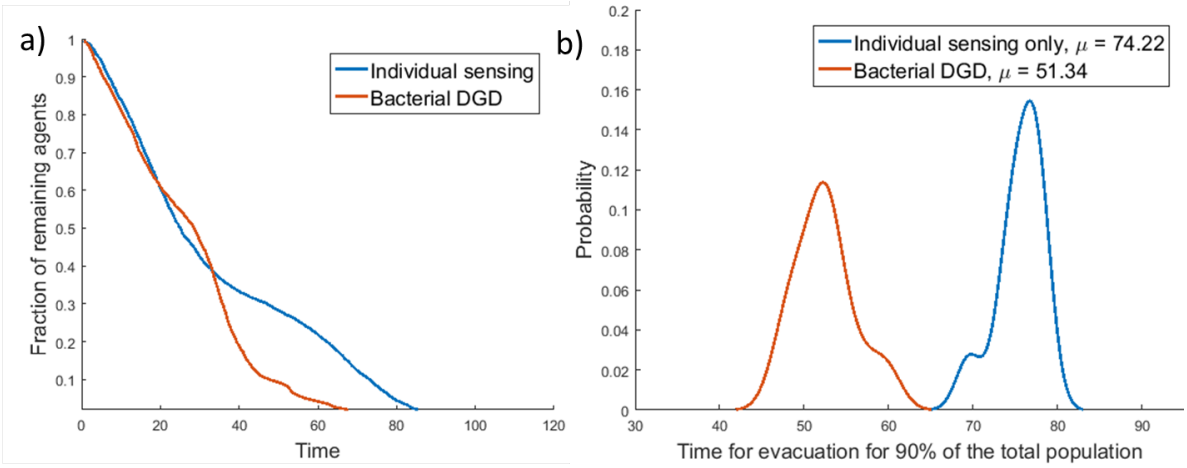

Figure 3.4: Simulation with the real floor plan of the 8^{th} floor of our departmental building with obstacle (see Fig. 3.3b). a) Plot of average fraction (over 10 independent trials) of remaining agents at any time. b) Distribution of required time to evacuate 90% of the total agents. In both the plots the performance of the bacterial DGD model with full communication (attraction and repulsion) is shown in comparison to individual sensing only. As we can see from the plots, with only individual sensing the model performs much worse than the proposed Bacterial DGD model.

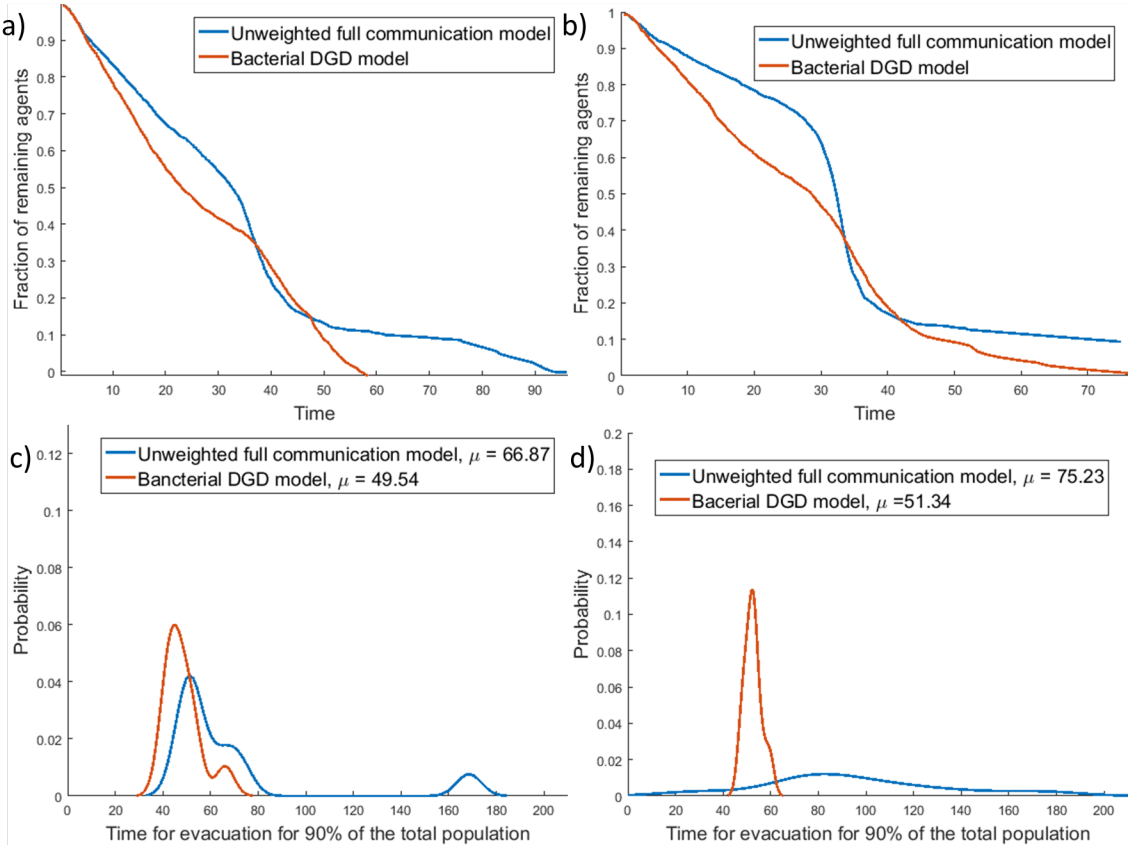

Figure 3.5: Simulation with the real floor plan of the 8^{th} floor of our departmental building with and without obstacle (see Fig. 3.3a-b). First column represents the without obstacle setup and the second column represent the with obstacle setup. a)-b) Plot of average fraction (over 10 independent trials) of remaining agents at any time. c)-d) Distribution of required time to evacuate 90% of the total agents. In both the plots the performance of the bacterial DGD model with weighted communication is shown in comparison to unweighted full communication model. As we can see from the plots, the unweighted full communication model performs significantly worse than the Bacterial DGD model in bothe setups. In the dynamic setup, the unweighted communication model fails to completely evacuate the floor.

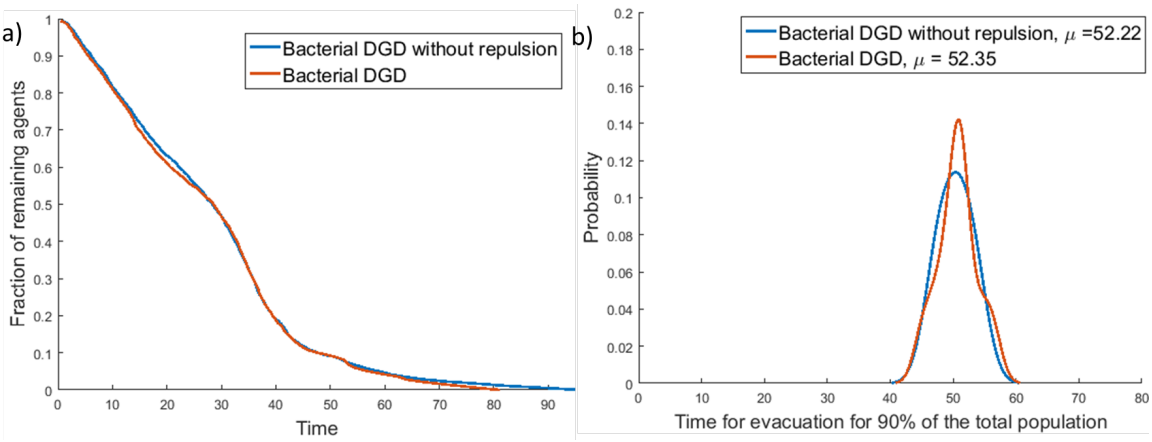

Figure 3.6: Comparison of Bacterial DGD model with and without repulsion. Simulation is performed on the real floor plan of the 8^{th} floor of our departmental building with obstacle (see Fig. 3.3b). a) Plot of average fraction (over 10 independent trials) of remaining agents at any time. b) Distribution of required time to evacuate 90% of the total agents. As can be seen from the plots, there is no significant difference between the performance of Bacterial DGD with and without repulsion.

component in the proposed Bacterial DGD evacuation model as well. To test whether accounting for repulsion causes any additional performance overhead, we have tested the DGD model with and without repulsion component. As can be seen from Fig. 3.6, we can see that the Bacterial DGD model with and without repulsion perform similarly, corroborating the fact that accounting for repulsion does not create any additional performance overhead.

To further investigate the efficacy of the Bacterial DGD model we have tested scenarios with multiple obstacles in the real floor plan of the building instead of just one obstacle. Fig. 3.7 illustrates the performance of Bacterial DGD model along with the other competing models in the real floor plan with 3 obstacles setup. As can be seen from the figure, the Bacterial DGD model outperforms other competing methods in this scenario as well. Moreover the difference of evacuation times for Bacterial DGD model and the other models further increases with addition of more obstacles.

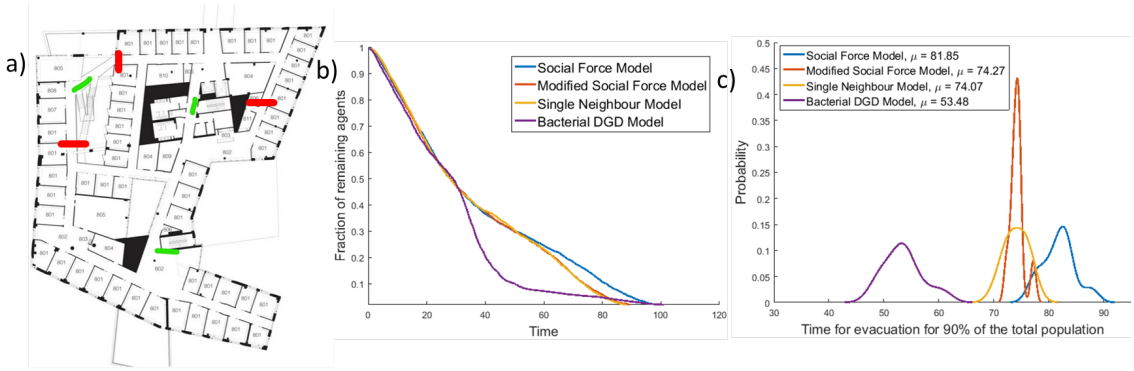

Figure 3.7: Simulation with the real floor plan of the 8^{th} floor of our departmental building. a) Floor plan with added obstacles. The green marks show the three exits of the floor. The red marks show the locations of the obstacles. b) Plot of average fraction (over 10 independent trials) of remaining agents at any time. c) Distribution of required time to evacuate 90% of the total agents. In both the plots the performance of the bacterial DGD model is shown in comparison to prior models, including social force, modified social force, and single neighbor model.

3.3.3 Using Euclidean distance instead of geodesic distance

Geodesic distance metric takes into account the topology of the environment which other point to point distance metrics do not. For example, two points may be very close in Euclidean space but are at two sides of a long corridor wall. In that case reaching from one to the other would take a long time in a realistic environment and this is captured by the geodesic metric. If we use Euclidean distance then the attraction force faced by the agents will only lead them towards the wall between them instead of exit. We thus believe that the best approach is to account for the structure of the building or topology of the environment when calculating distances. We have compared the performance of both Euclidean distance metric and geodesic distance metric. As can be seen from Fig. 3.8, use of Euclidean distance metric significantly increases the evacuation time as compared to the geodesic distance metric.

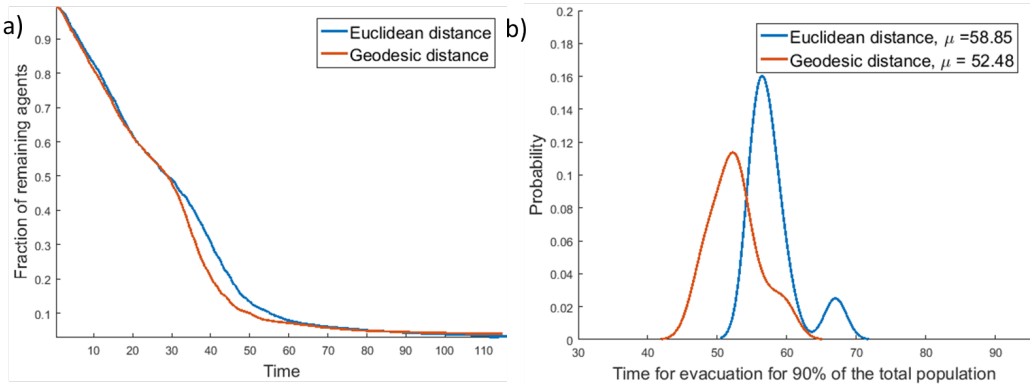

Figure 3.8: Comparison of Bacterial DGD model with geodesic and Euclidean distance metric. Simulation is performed on the real floor plan of the 8^{th} floor of our departmental building with obstacle (see Fig. 3.3b). a) Plot of average fraction (over 10 independent trials) of remaining agents at any time. b) Distribution of required time to evacuate 90% of the total agents. We can see from the plots that, Euclidean distance metric performs significantly worse than the geodesic distance metric.

3.3.4 Sensitivity analysis of Bacterial DGD hyperparameters

There are seven hyperparameters of the Bacterial DGD model, RR: Radius of repulsion, C_a and C_b are the coefficient of the exponential weighting for location and speed messages, k_γ is the coefficient of the exponential weighting of individual variable speed, L and T are the parameters of the discretization function, L is the number of possible messages, and T, an upper bound above which all messages as treated as the highest value possible (see [56], and σ_J is the variance of the Gaussian noise added to the gradient estimation . For both the basic room and real floor plan setup we have used same set of hyperparameters ($RR = 0.4, C_a = 2, C_b = 4, k_\gamma = 0.01, L = 8, T = 1, \sigma_J = .01$). To analyze the sensitivity of the method to the hyperparameters, we have run the Bacterial DGD model in the real floor plan with obstacle setup (Fig. 3b) with varying values of the hyperparamters. Table 3.1 report the Mean Evacuation Time (MET) (over 10 independent trials) to escape 90% of the agents. We can see that the most sensitive parameter is k_γ, increasing it increases the evacuation time and vice versa. The parameter directly controls the

variable step size of the individuals. Hence, such effect is expected. We have set this parameter such that the average step size over the course of the simulation is equal to the step size used in the comparing methods. The model was relatively robust to changes in other hyper parameters.

To see how the proposed algorithm compares to the optimal solution, we first define the optimal time to evacuate as the ratio of maximum exit distance to the average step size. Without the presence of obstacle in real floor plan (Fig. 3.3), the optimal time to evacuate is 38 simulation rounds, whereas with bacterial DGD the evacuation time is 48 rounds. As mentioned earlier, the increased evacuation time can be attributed to the communication overhead of the DGD model. To see whether perfect communication (without noise) would result in an optimal solution, we simulated the bacterial DGD model without adding any communication noise. In the perfect communication model, the evacuation time is 43 rounds, which is still longer than the optimal solution, but the difference is less pronounced. In the setup with dynamic obstacle addition, the optimal evacuation time is 47 rounds which is much closer to the performance of the bacterial DGD, showing that in the presence of dynamic obstacle the bacterial DGD performs closer to the optimal solution. When we allow perfect communication the evacuation time of bacterial DGD is 50 rounds, more closer to the optimal solution.

3.4 Discussion

We proposed a method which extends the bacterial based Distributed Gradient Descent (DGD) model for mass scale evacuations. Agents combine their own knowledge with information communicated by other agents. The model uses simple messages, does not require knowledge of the number and identity of the other agents and only uses simple computations making it ideal for constrained settings.

We have shown, both theoretically and in simulations, that the DGD method leads to convergence of interacting agents. We performed simulation analysis comparing the DGD method to prior methods in both static settings and dynamic settings when new obstacles are introduced.

Table 3.1: Sensitivity analysis of Bacterial DGD hyperparameters

L	T	RR	C_a	C_b	k_γ	σ_J	MET
8	1	0.4	2	4	.01	0.01	53
8	1	0.4	2	4	.05	0.01	67
8	1	0.4	2	4	.005	0.01	42
8	1	0.4	2	2	.01	0.01	56
8	1	0.4	2	6	.01	0.01	52
8	1	0.4	1	4	.01	0.01	51
8	1	0.4	3	4	0.01	0.01	53
8	1	0.2	2	4	0.01	0.01	54
8	1	1.0	2	4	0.01	0.01	53
8	2	1.0	2	4	0.01	0.01	54
8	.5	1.0	2	4	0.01	0.01	52
6	1	1.0	2	4	0.01	0.01	54
10	1	1.0	2	4	0.01	0.01	55
10	1	1.0	2	4	0.01	0.02	57
10	1	1.0	2	4	0.01	0.03	59

As we have shown, there is a (small) price that we pay when using the DGD model if the environment does not change, primarily due to bottlenecks caused by the attraction terms. However, in cases where obstacles or other constraints are dynamically added we observe a very large improvement when using the (simple) bacteria based computation model. Aggregate information from a small set of agents who have found a route to the exit can quickly lead to fast evacuations of all individuals. Specifically, in the case of a blocked door or corridor the DGD method reduces evacuation time by roughly 40% when compared to all prior methods, which could have a large impact in cases of real emergencies. The reduction in evacuation time further increases when more obstacles are present in the environment (50% for 3 obstacles).

The DGD model improves performance even though it relies on simpler assumptions than prior models. For example, Bacterial DGD does not assume identifiability of the communicating agents. In contrast, for prior methods such as leader based evacuations [34] identifiability is a crucial requirement. A potential issue with leader based strategies is the dynamic nature of the obstacles we assume in our studies. Even though the leader might know more about the static environment, s/he will be completely unaware of the dynamic obstacles. Hence following the leader can result in congestion at the obstacle site and subsequently increase the evacuation time.

Chapter 4

Adaptive Tumbling in Bacterial Chemotaxis on Obstacle-laden Terrains

4.1 Introduction[1]

While the DGD model presented in Chapters 2 and 3 performs well on simulated and computational settings, they have not been tested on real biological data. In this chapter we discuss such validation experiments in which we designed and fabricated environments with obstacles to test the performance of several proposed chemotaxis models, including DGD. As noted in the Introduction, the mechanisms of bacterial chemotaxis have been extensively studied for several decades, but how the physical environment influences the collective migration of bacterial cells remains less understood. Previously developed models for collective bacterial behavior do not take into account the impact, and feedback, from the physical environment on the cell state [25, 26, 27, 28]. For example, when bacteria traverse obstacle-laden surfaces, current models indicate that exit times would increase with increased obstacle coverage since obstacles limit the ability of cells to select an optimal route. To test if this is indeed the case, we performed experiments in which we studied the movement of *E. coli* cells in microfluidic chambers containing

[1]This chapter is based on the paper [72]

physical obstacles. We varied the obstacle size or the overall surface coverage by obstacles. Contrary to predictions of current models, we found that average bacterial exit times remain nearly constant regardless of the specific obstacle coverage or size. Following adjustments to our bacterial DGD models, we found that two changes in the model can explain the observed behavior. The first is a change in the communication protocol that limits the sending of messages (secreted signaling molecules) to cases where the cells observe an improvement in the gradient. The second modification was adaptively changing the tumbling rate based on the presence and coverage of obstacles. We tested the latter prediction both experimentally and in computational simulations and found that higher obstacle density indeed leads to adaptively lower tumbling rates over time. These findings imply operational short-term memory of bacteria while moving through complex environments in response to chemotactic stimuli and motivate improved algorithms for self-autonomous robotic swarms.

In addition to these mathematical models developed to explain chemotaxis, this process also serves as the basis for several distributed computing swarm based methods. For example, robotic swarm methods for searching for trapped victims in emergency situations are often based on chemotaxis [73],[74]. In such applications, robots integrate signals from victims (either based on smell or sound) with visual information from neighboring robots to determine their search route. We show that by incorporating adaptive tumbling strategy of bacteria, these robotic swarm algorithm actually improves their search performance in complex environment.

4.2 Computational methods

4.2.1 Mathematical and computational models for bacterial chemotaxis

We tested several models previously developed to explain bacterial chemotaxis including those that focus on individual cells and those that consider communication between cells. Brief descriptions of the models are presented next.

Keller-Segel model [25]

The Keller-Segel model describe the density distribution in space and time of bacteria cells as function of the concentration gradient of a given chemoattractant. Formally,

$$\frac{\partial b}{\partial t} = \mu \frac{(\partial^2 b)}{(\partial^2 x)} - \delta \frac{\partial}{\partial x}(b(\frac{\partial ln(s)}{\partial x}) \tag{4.1}$$

Here $b(x,t)$ is the cell density as function of x and t, s is the chemoattractant concentration, μ and δ are constants. To simulate this model, we made the movement direction of each cell a function of only the chemoattractant gradient, such that if the cells are moving towards the positive gradient they will reduce their direction change, thus following the gradient. None of the neighboring cells affect the movement direction of the particular cell.

Individual chemotactic movement model based on Brown-Berg experiment [28, 75]

In this model, it is assumed that a cells trajectory is made up of a series of runs and tumbles. In run, the cells move in a constant speed v at a direction p. The probability of a cell terminating the run by tumbling in the time interval $(t, t + \Delta t)$ for small δt is $\lambda \delta t$, where λ is the stopping rate. λ is reduced when a run is up the chemotactic gradient and increased when a run is down the gradient. In general, λ will vary nonlinearly with $p.y$, the component of swimming direction parallel to the concentration gradient, however for relatively weak chemotaxis, the dependence of λ on these quantities can be linearized. To simulate this model, we made the duration of trajectory a function of the perceived gradient. In this way the cells can follow the chemical gradient.

Shklarsh model [26]

Unlike the previous models, the DE model proposed by Shklarsh et al. [26] allow the cells communicate with each other to efficiently respond to an attractant (search for food) through chemotaxis. In this model, each cell can communicate with its neighbors in three possible ways,

1. repulsion
2. orientation
3. attraction

within a fixed sphere of influence and the selection of communication type depends on the distance between the two communicating cells. The repulsion, orientation, and attraction zones are three concentric circular zones with fixed radii (B_{RR}, B_{RO}, B_{RA},and $B_{RR} < B_{RO} < B_{RA}$) centered at the current location of the cell. Lets consider the distance between two cells i and j is d_{ij}. The rules of communication are as following:

- Repulsion: If $d_{ij} < B_{RR}$, the cells will repel each other
- Orientation: If $B_{RR} \leq d_{ij} < B_{RO}$, the cells will orient themselves in the same movement direction
- Attraction: If $B_{RO} \leq d_{ij} < B_{RA}$, the cells will move towards the other cell

A key issue with this model is that it does not consider anonymity of the senders. In this communication scheme, each cell has to know the exact location of the communicating cell to properly place within the appropriate communication zone. In addition to that, all the messages from a zone are weighted equally without considering distance between cells. This is a biologically unrealistic assumption. Since cells communicate with each other via secretion, the strength of messages decay as the distance between cells increases. The DGD model discussed next overcomes these limitations of the DE model.

4.2.2 DGD model for bacterial chemotaxis

In the DGD model the movement of an agent i at time n is a function $d_i(n)$ of two quantities: its own sense of direction $\theta_i(n)$ (based on the chemical gradient), and the locations and movement directions of other migrating cells. In this dynamic interaction network of migrating bacteria, nodes correspond to bacteria, at some current location $x_i(n)$, and edges, representing

physical distance, exist between every pair of cells. The DGD model updates $d_i(n)$ based on individual belief and the beliefs of neighbors, while using simple messages (formalized below). The model assumes that individual cells follow a chemical gradient of food source by decreasing their (random) tumbling angle at high concentrations and thus largely move in the direction of the attractant. Specifically, bacteria perpetually move in a direction that they repeatedly perturb randomly. The magnitudes of these perturbations are inversely related to the change in the attractant concentration between iterations, with the approximate effect that the agent (i.e., the bacterial cell) continues to move in gradient direction. Formally, under these assumptions, at time n, cell i changes its direction by an angle $\theta_i(n)$, which is a function of $c_i(n)$, the difference in food concentration between the current and previous time steps. Specifically, the new tumbling angle $\theta_i(n)$ is sampled randomly from a Gaussian distribution $\theta_i(n) \sim N(\theta_i(n-\Delta n); \sigma(\Delta c_i(n))^2)$ centered at the previous angle $\theta_i(n-\Delta n)$, with the variance $\sigma(\Delta c(n))^2$ given as:

$$\sigma(\Delta c(n))^2 = \begin{cases} 0; & \Delta c(n) \geq 0 \\ \pi; & \Delta c(n) < 0 \end{cases} . \tag{4.2}$$

Thus, based only on its own perception of the food gradient, the i^{th} agent updates its location $x_i(n)$ according to

$$x_i(n+\Delta n) = x_i(n) + s_i(n)\Delta n \tag{4.3}$$

where $s_i(n)$ is the unit vector in the direction of the movement.

$$s_i(n) = (cos(\theta_i(n)); sin(\theta_i(n))) \tag{4.4}$$

The above equation is based on individual sensing only. However, in most cases bacteria move in large groups where the cells have the ability to communicate with each other. They communicate via secreting signaling molecules that can be detected by the other neighboring cells [18, 19]. Through these molecules the cells can send information about their current location. However, these molecules also diffuse over the terrain, meaning the cells closer to the sender will sense a stronger message compared to cells that are farther away. Then each cell

combines the messages from all other cells to determine the next direction of movement. Let $u_i(n)$ denotes this communication component of the model. At each step $u_i(n)$ is updated as following:

$$u_i(n) = D_{L,T}\left(\sum_j \frac{\exp(-(C_a\|x_i(n) - x_j(n))\|)(x_i(n) - x_j(n))}{\|x_i(n) - x_j(n)\|}\right). \tag{4.5}$$

$D_{L,T}$ is a discrete thresholding operator parameterized by L, a positive integer denoting the number of possible messages, and T, an upper bound above which all messages are treated as the highest value possible (see [56] for the exact construction of this stone-age computing" threshold, which has been used in ant migration models). C_a is a positive diffusion constant, determining how quickly the location signal diffuse from the source agent. Under this model, bacteria communicate the attraction information using only $log2L$ bits.

The model also includes a physical repulsion term. If an agent physically interacts with another agent it moves in the opposite direction to where it came from regardless of the other information it has. Hence the direction will be updated, as follows:

$$u_i(n) = -\sum_{j \in B_{RR}} \frac{x_i(n) - x_j(n)}{\|x_i(n) - x_j(n)\|}. \tag{4.6}$$

Where RR is an influence radius that is related to the physical size of the agents and their speed such that agents cannot reach agents outside of RR in a single round [26]. Finally, the agent combines the messages it received with its own observation $s_i(n)$, resulting in the following modification of equation 4.3:

$$x_i(n + \Delta n) = x_i(n) + (\frac{u_i(n)}{|u_i(n)|} + ws_i(n))\Delta n \tag{4.7}$$

w is a scalar constant. A schematic layout of the original DGD model is presented in Fig. 4.1.

4.2.3 Revised DGD model

When we compared the performance of aforementioned models on different environmental conditions with experimental data, we observed some contentions. The experiments show that the

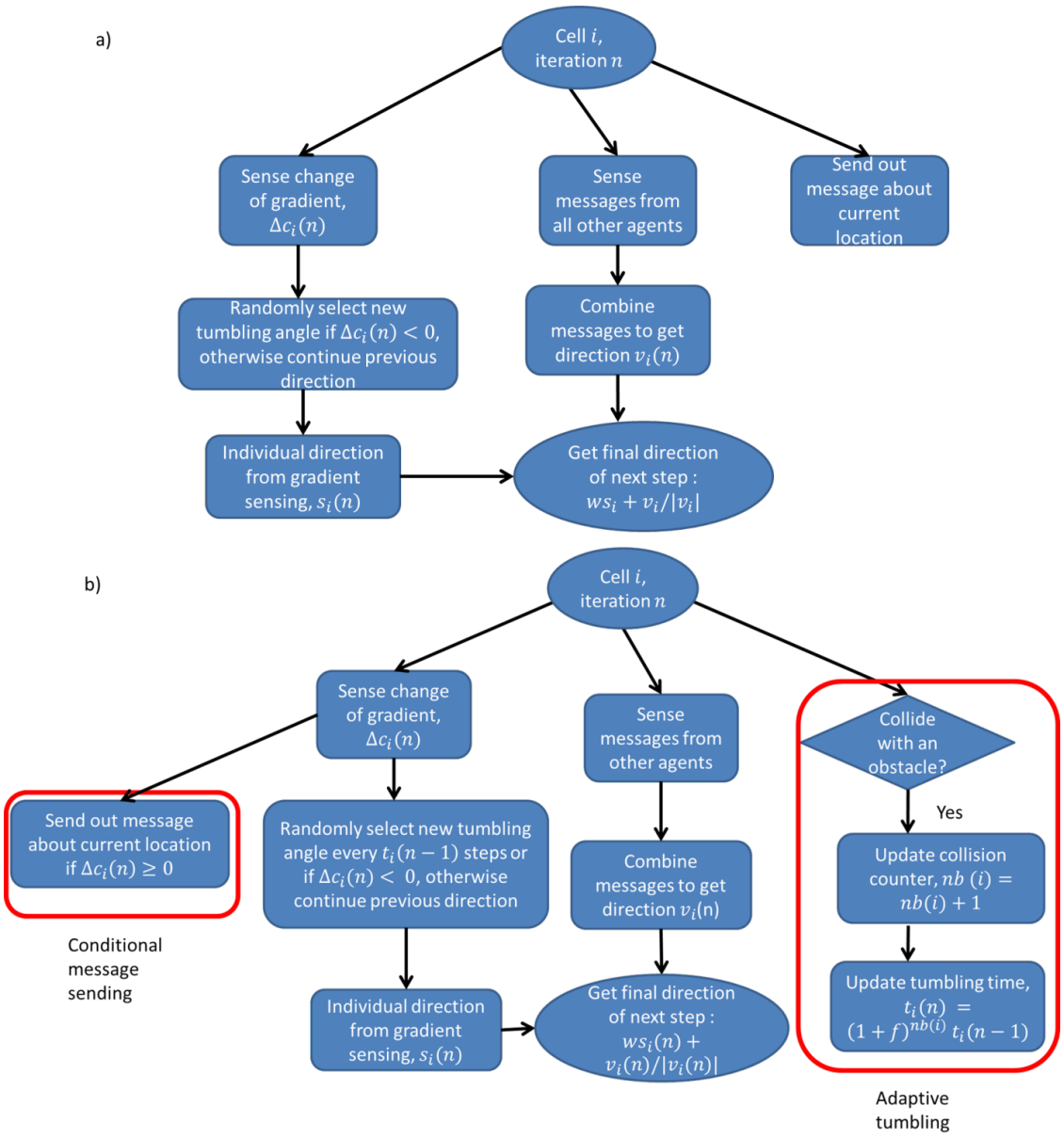

Figure 4.1: Layout of a) Original b) Revised DGD model. The layout shows the actions taken by a cell in each iteration to determine the direction of the movement. The original model combines individual gradient sensing with combined messages from other agents to determine the direction of next step. However, the original model did not account for the environmental effect on bacterial movement. Hence, in the revised model we have accounted the effect of the environment by introducing adaptive tumbling. In addition to that, we have introduced conditional message sending, i.e., cells can only send messages when they are moving in the correct direction. These two changes are highlighted by red boxes in the revised model layout.

migration time of cells do not vary with the obstacle coverages, while in simulation of the models the migration time increases with the increase of obstacle coverage (Results). To better model the bacteria chemotactic movement and produce coherent results with the experiments, we further revised the proposed bacterial DGD model. The details of the revised model is presented below.

Chemotaxis models [26],[27] assume that the cells communicate via chemical secretion at all points of their search, irrespective of their individual sensing of the attractant gradient. However, recent observations suggest that this may not be the case. Instead, cells only signal (via secretion of amplifying signaling molecule(s)) when they are moving in the right direction (i.e., positive gradient change) [20]. We have thus revised the model so that only messages from cells with positive gradient change are allowed. The gradient change perceived by cell i is denoted by $\Delta c_i(n)$. We update eq 4.5, as follows:

$$u_i(n) = D_{L,T}\left(\sum_{j,\Delta c_j>0} \frac{\exp(-(C_a\|x_i(n)-x_j(n))\|)(x_i(n)-x_j(n))}{\|x_i(n)-x_j(n)\|}\right). \tag{4.8}$$

The revised model also allows cells to adjust their tumbling frequency according to their perception of the environment. We assume that cell i iteratively updates (via a feedback loop) its tumbling time, denoted t_i, as follows (note that t_i is tumbling time in seconds, tumbling rate is defined as $1/t_i$).

$$t_i(n) = (1+f)^{nb_i(n)} t_i(n-1) \tag{4.9}$$

Here, $nb_i(n)$ is the number of times agent i has encountered an obstacle before iteration n. f is a constant set at 0.0005. This adaptive formulation of t_i change throughout allows the cells to adjust their movement in high obstacle coverage. Using the new rate we update eq 4.2 as following:

$$\sigma(\Delta c_i(n))^2 = \begin{cases} 0; & \Delta c_i(n) \geq 0 \& \delta_i(n) < t_i(n) \\ \pi; & \Delta c_i(n) < 0 || \delta_i(n) \geq t_i(n) \end{cases}. \tag{4.10}$$

Here $\delta_i(n)$ denotes the number of iterations since the last tumble for cell i at iteration n.

Finally, we model variable cell sensing abilities (corresponding to difference within cell pop-

ulations) by adding an additional Gaussian noise term to $\Delta c_i(n)$:

$$\Delta c_i(n) = N(\Delta c_i(n); \Delta c_i(n)(1 - \alpha_i)) \tag{4.11}$$

Here α_i denotes cell i's specific sensing ability, which we assume varies between 0 and 1 (0 denotes the lowest sensing ability, i.e., cells cannot perceive the food gradient at all). A schematic layout of the revised DGD model is shown in Fig. 4.1.

4.3 Experimental methods

4.3.1 Bacterial strains and growth conditions

The wild-type *E. coli* K12 strain RP437 that constitutively express yellow fluorescent protein (YFP) from the plasmid PZA3R-YFP or mCherry from the plasmid PZA3R-mCherry, both of which containing chloramphenicol resistance, were used. Strains were first grown overnight at $30°C$ with strong agitation (240 rpm) in M9 minimal medium supplemented with $1g/L$ casamino acids and $4g/L$ glucose (M9CG) and appropriate antibiotics. Cultures were then diluted 100 fold in fresh M9CG and grown at $30°C$ until early exponential phase, optical density at $600nm$ (OD600) of 0.1-0.2. Prior to loading the bacteria into the microchambers the cultures were centrifuged, washed, and resuspended in fresh testing medium. Testing media used were either M9CG or M9G (M9 minimal medium supplemented with $4g/L$ glucose and $500g/mL$ each of L-threonine, L-leucine, L-histidine, L-methionine, and thiamine).

4.3.2 Microfluidic device

We designed and fabricated a microfluidic device (Fig. 4.6) containing eleven consecutive microchambers that were connected to a wide channel through a 5μm-wide and 40 μm-long inlet channel to allow for the introduction of *E. coli* cells and media into the microchambers. The microchambers were 1 mm wide and 1 mm long (excluding the nook area) and contained evenly

distributed micropillars. The microdevice was fabricated by standard soft lithography, as we described previously [19]. Briefly, 10μm-thick photoresist SU-8 2010 (MicroChem, Newton, MA) was spin-coated onto a clean silicon wafer. The wafer was then exposed to UV light through the photomask by Karl Suss MJB3 aligner. After removing the unexposed resist in SU-8 developer, the resulting positive reliefs of the microchannels and the microchambers on the wafer served as a master mold. A 10:1 w/w mixture of the silicon elastomer PDMS prepolymer and its curing agent (Sylgard 184 set, Dow Corning) was then poured over the mold and cured at 65C overnight. After curing, the PDMS replicates were peeled off from the mold and bonded to a glass slide by brief treatment in air plasma. Immediately after the plasma treatment and bonding, 5 μL of $20mg/mL$ BSA solution was added into the two inlet/outlet holes of the PDMS device and kept at room temperature for one hour to coat the walls of the PDMS device with a thin layer of BSA to prevent cell adhesion. We then replaced the BSA solution in the microchannels and microchambers with one of the testing media prior to loading the cells. 10μM L-aspartic acid (Asp) was usually added to the testing medium to facilitate the cell loading into the microchambers. Motile *E. coli* cells that were resuspended in fresh testing medium (without Asp) were then introduced into the main channel and allowed to swim into the microchambers continuously. After reaching the desired cell density in the microchamber fresh testing medium containing 200 μM aspartate as chemoattractant was pumped through the main channel at a low rate of 5μL/min using a syringe pump (Pump Systems Inc.).

4.3.3 Time-lapse imaging

The migration of YFP-expressing *E. coli* cells in the microchambers were observed and recorded in fluorescence mode using a fully automated inverted microscope (Zeiss AxioObserver Z1), equipped with a motorized x-y stage (Applied Scientific Instruments). Time-lapse movies were acquired at a rate of 2 frames/min in population level experiments and 3 or 10 frames/second for 10-15 minutes in single cell tracking experiments using a CCD camera (Zeiss AxioCam MRm)

at room temperature ($26^{\circ}C$). The movies were processed for analysis with ImageJ software. The cells inside the microchambers in each image were counted automatically by a custom pipeline with the opensource software, CellProfiler [76]. For the analysis of the movement of individual cells, tracking of each cell was performed with Trackmate, a Fiji plugin for single particle-tracking. A detection algorithm that applies a plain Laplacian of Gaussian filter on the image (LoG detector) was used. All calculations were made in the Fourier space. A 3×3 median filter that filters out spurious spots and the built-in sub-pixel localization algorithm were applied to optimize spot detection. The thresholds for size and brightness were adjusted for each movie individually, such that all the cells were included. The simple Linear Assignment Problem (LAP) tracker, which was first developed by Jaqaman et al. [77] was the particle-linking algorithm used to track the spots. The principle of the algorithm is linking a cell in a frame with the closest cell in the next frame within a maximum distance, which was set to 12.5μm. This simple LAP tracker allows for gap-closing events, in which a spot may disappear (because it moved out of focus) and reappear later but does not allow for splitting or merging of tracks. In instances where there are gaps, the gap closing maximum distance was set to 25 μm, and a maximum of 5 frame gaps were allowed. Experimental data is available in the supporting website.

4.4 Analyzing single cell tracks

We also generated single cell tracks under different obstacle coverages. To quantitatively analyze the tracks, we computed two quantities, one is tumbling frequency by detecting individual tumbling events from the tracks and the other is mean square displacement (MSD) values as a function of time. By fitting an exponential curve to the MSD plot we get a global estimate of critical time of tumbling. These methods are discussed in details in the next sections.

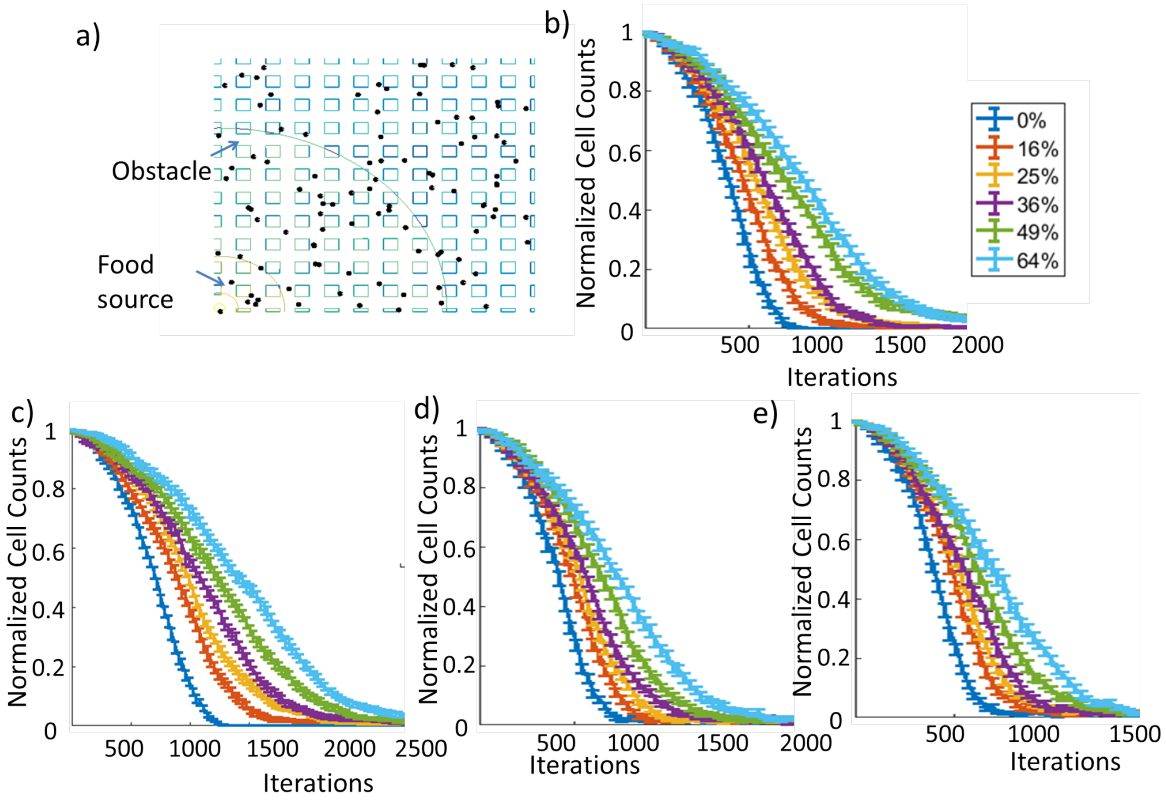

Figure 4.2: Simulation results of current chemotaxis models. We tested several different bacterial chemotaxis models simulating cells faced with varying obstacle coverage within an attractant gradient. All models predicted much slower escape times when the environment contains more obstacles. (a) Terrain model for simulations of bacterial food search. Obstacles are placed in a grid and the attractant source is at the bottom right corner, as in the experimental setup. Black dots denote bacteria (agents) (b) Adaptive step size model based on Brown-Berg experiment [28, 75] (c) Keller-Segel model [25] (d) Shklarsh model [26] (e) DGD model [27]. It is evident, that all models predict the fastest exit rate of cells in the absence of any obstacles.

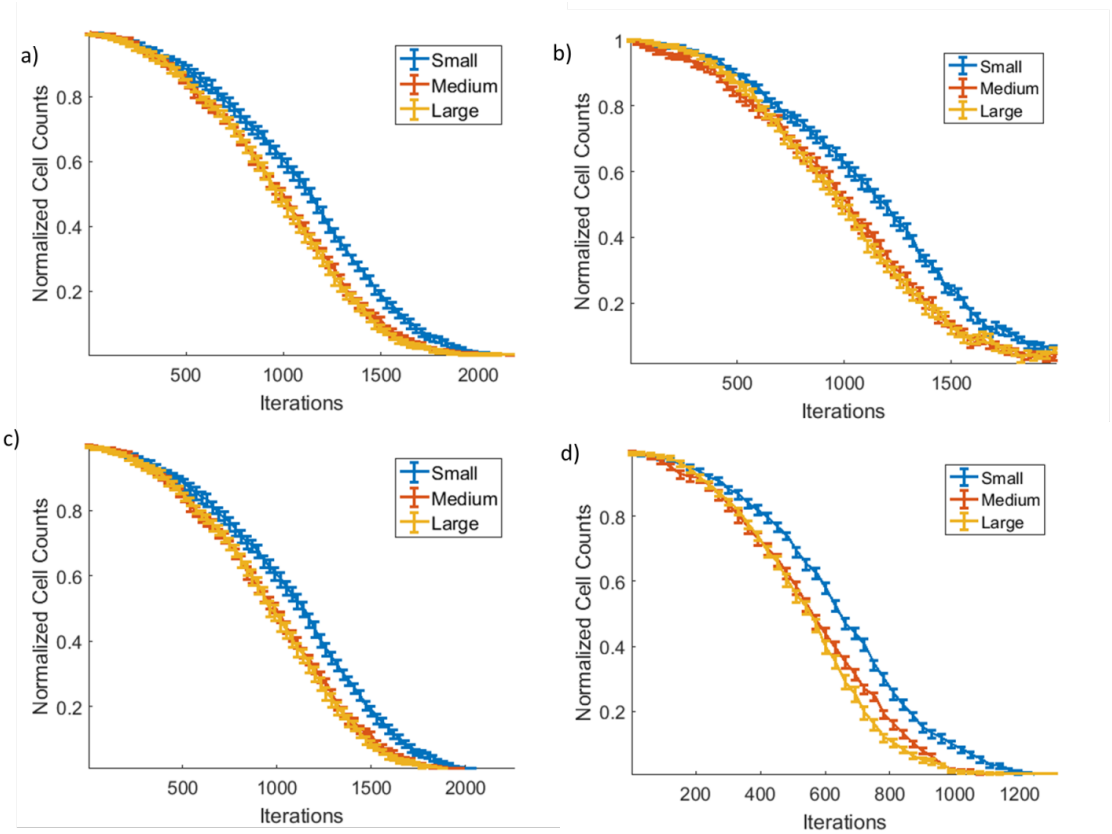

Figure 4.3: Different bacteria migration chemotaxis model performance under varying obstacle sizes. (a) Adaptive step size model based on Brown-Berg experiment [28], [75] (b) Keller-Segel model [25] (c) Shklarsh model [26] (d) DGD model [27]. As can be seen, all models predict faster times in small obstacle size.

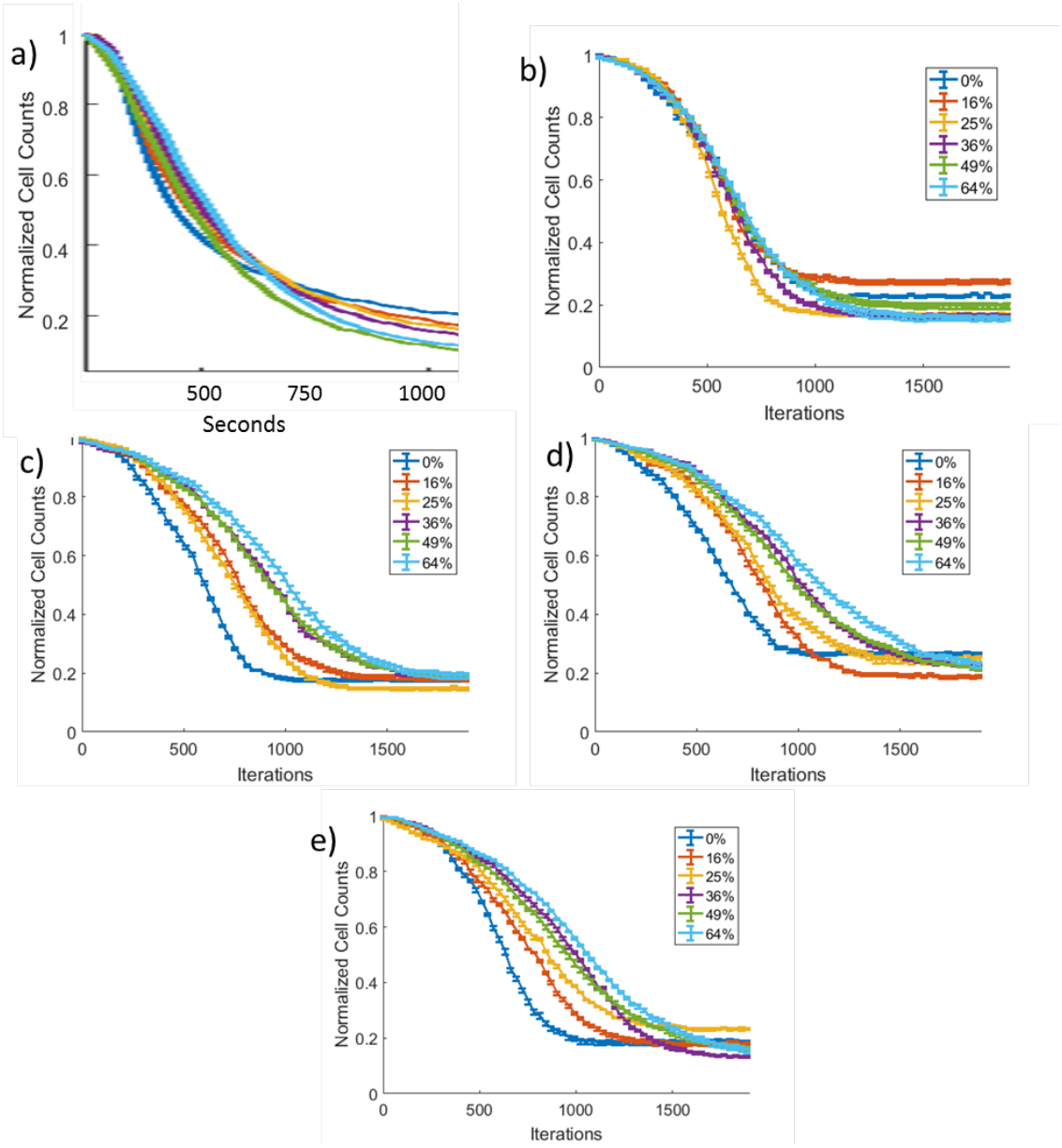

Figure 4.4: Simulation results of the revised bacterial DGD model and prior chemotaxis models with approximately 20% immobile cells. We tested several different bacterial chemotaxis models simulating cells faced with varying obstacle coverage within an attractant gradient. All models predicted much slower escape times when the environment contains more obstacles. (a) Experimental results (b) Revised DGD model (c) Adaptive step size model based on Brown-Berg experiment [28, 75] (d) Keller-Segel model [25] (e) Shklarsh model [26].

4.4.1 Computing tumbling rates

To compute tumbling frequencies (TFs) we first define a Tumble when the angular movement of a cell reaches a certain threshold (80° in this this, results are robust to the selection of the threshold). Since cells complete a tumble over a period of time, we analyze multiple consecutive frames (frame rate: $10/3$ frames per second) to detect a complete tumble. The formal algorithm to detect a tumble is given in Algorithm 1. We take an iterative approach to count the total number of tumbles in a single cell track. From a point of origin in the track we compute cumulative angular movement in the subsequent points. As soon as the cumulative angular movement is above a certain threshold, we define it as a tumble and update the point of origin at the detected tumble location. We again compute cumulative angular movements and so forth until the cell exits or the movie ends. Due to noisy measurement we sometimes observe multiple large angular movements in a very short time span that do not represent actual tumbles and falsely increase the total tumble counts. To avoid that, for each detected tumble we investigate whether a tumble was detected in previous Nc (where we set $Nc = 3$ here) consecutive frames. If we observe multiple tumbles within Nc consecutive frames, we only consider the tumble with the maximum angular displacement.

4.4.2 Mean Square Displacement (MSD) analysis

Next we computed MSD to get a global estimate of tumbling rate. MSD as a function of time exhibits 2 distinct regimes. In the short time scale it exhibits a power-law behavior ($x^2 \propto t^\alpha$), with power $\alpha = 2$, which reflects the ballistic nature of the motion at that scale. In the long time scale the MSD exhibits a diffusive behavior, where x^2 is proportional to t. The transition time between these two regimes t_c is thus the average lifetime of the ballistic motion, i.e. it is the average time of a straight run or the inverse of the tumbling frequency [78]. We quantify the time between tumbles (t_c) by fitting an exponential curve $4Dtexp(-t/t_c)$ to the log-log MSD plots, where D is the diffusion constant. This equation describes the Brownian diffusion process

of molecules [26].

Algorithm 1: Algorithm for tumble detection

Result: Tumbling frequency, T_f

Total frame number = N;

Frame rate per seconds = f_p;

initialization;

Tumble detection threshold = h ;

Number of comparison points = Nc;

Tumble indices, T_i = [];

Origin, s = 1;

Current frame, i = 2;

while $i < N$ **do**

    Vector between points i and s: p;

    Vector between points i and $i + 1$: q;

    Current angle ϕ_i = $angle(p, q)$;

    if $\phi_i \geq Th \& \{(i - Nc), ..., (i - 1)\} \notin T_i$ **then**

        $s = i$;

        $T_i = append(Ti, s)$;

    end

    else if $\phi_i \geq Th \& \{(i - Nc), ..., (i - 1)\} \in T_i$ **then**

        $s = argmax(\phi_i : i \in (i - Nc), ..., i)$;

        $T_i(\{(i - Nc), ..., (i - 1)\} \in T_i) = []$;

        $T_i = append(T_i; s)$;

    end

end

Total number of detected tumbles, $Nt = |T_i|$;

Tumbling frequency,$T_f = Ntf_p/N$;

4.5 Results

4.5.1 Analysis of existing chemotaxis models

To determine the predicted behavior of bacterial cells when faced with physical obstacles we simulated several topologies with regularly spaced obstacles (Fig. 4.2a). We next tested several existing models of bacterial chemotaxis and examined their predicted exit time (time to reach the attractant source) in such settings. These include the Keller-Segel model [25] and models based on Brown & Berg [28, 75], which account only for individual cell movement in response to chemoattractant. Another model we tested was a differential equations (DE) model by Shklarsh et al [26] that divides the area around each cell into three different compartments and allows for both physical and chemical communication between cells, where depending on their distance, neighboring cells either attract or repel each other (see Methods). Finally, we considered our previously developed DGD model that allows for a discrete set of messages to be exchanged among cells and for cells to combine their internal sensing (based on the gradient they observe) with the messages received from other cells to determine their next move [27]. Figure 4.2b-e depicts simulation results for these models on environments containing different numbers of regularly spaced obstacles of the same size. In these simulations, the area covered by the obstacles ranges from 0% to 64%. As can be seen, all models predicted that with 0% coverage (no obstacles) bacteria would reach the attractant source faster than in the presence of increasingly higher obstacle coverage.

4.5.2 Experimental results contradict model predictions

Given the simulation results, we next asked if these models can indeed qualitatively describe the collective migration behavior of *E. coli* cells in complex environments. In addition to model validation, such experiments may provide new insights and can help further improve algorithms and analysis [79, 80]. To this end, we designed and fabricated a microfluidic device to test the

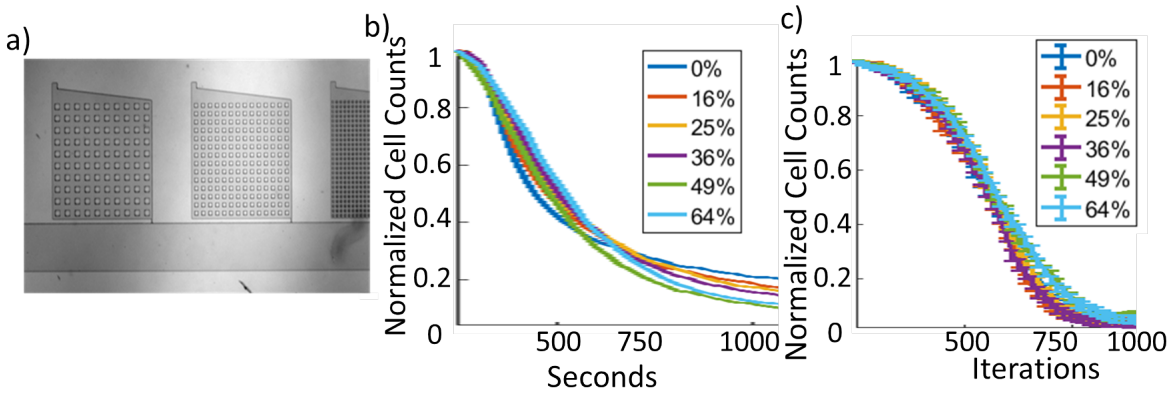

Figure 4.5: Experimental results and revised DGD model. We performed experiments to test the accuracy of the predictions made by all models. (a) Design patterns of a microfluidic device with different obstacle size. Other designs tested include those with the same obstacle size but different coverage (See Fig. 4.6). (b) Total cell count of YFP-labeled, wild-type *E. coli* RP437 cells in the microchambers as a function of time, after applying a M9CG chemotaxis stimulus. Note the difference between the experimental results and simulation results from Fig. 4.2 . (c) Simulations using the revised DGD model for the same settings. Changing the secretion model and adjusting tumbling rates based on the environment, the updated DGD model leads to results that are concordant with the experimental results.

chemotaxis of *E. coli* cells in terrains with and without obstacles (Figs. 4.5, 4.6, Experimental Methods). These devices allowed us to test the ability of migrating bacteria to navigate in microchambers with different obstacle sizes and coverage.

We first performed experiments in which we followed cells in environments with 0%, 16%, 25%, 36%, 49% and 64% of total area covered by obstacles of identical size ($50\mu m^2$) (Fig. 4.5a). We used either a complex medium (M9CG) (Fig. 4.5b), where the attractant gradient was created by the bacteria depleting the nutrients in the rectangle chamber while fresh nutrients are supplied through one of the rectangles corners, or a single attractant medium (200 μM aspartate [Asp] in M9-G), where 200 μM aspartate was supplied continuously through one of the chambers corner.

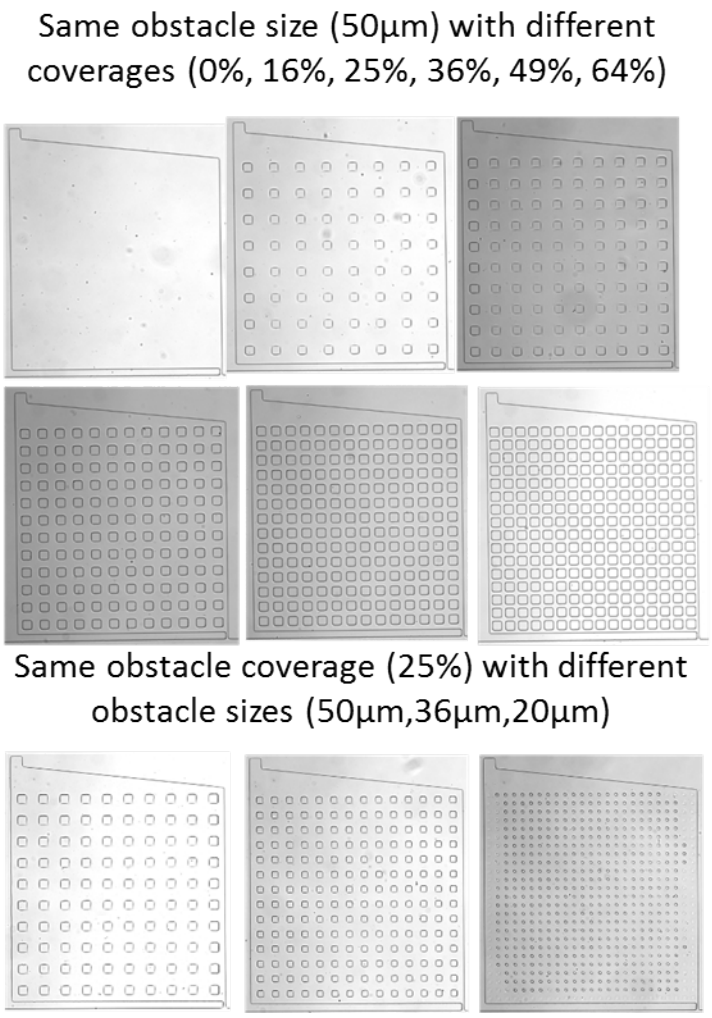

Figure 4.6: Different microchamber setups. The top panels depict microchambers with no obstacles (0%) or various total surface coverage with obstacles of identical sizes ($50\mu m^2$). The bottom panels shows microchambers with various denoted sizes with a total of 25% surface coverage.

(Fig. 4.7). The gradient profile is shown in Fig. 4.8. To determine the impact of the various obstacle types on the time it takes *E. coli* cells to reach the attractant source, we measured the change in the fraction of cells remaining in the chamber as a function of time. In contrast to predictions by all models (Fig. 4.2b-e), *E. coli* cells displayed no substantial difference in their average exit times between different overall coverages. In fact, regardless of the overall surface coverage *E. coli* cells exited from the microchambers at nearly identical rates. We also tested the migration of cells using microchambers containing different obstacles shape (round instead of square) with the same surface coverage as before and observed the same results, as well as different obstacle sizes but constant total surface coverage (25%). We again observed that variation in obstacle size and shape does not affect the bacterias chemotactic migration towards the attractant source (Figs. 4.9, 4.10).

4.5.3 Performance of revised DGD model

Given the disagreement between experimental and simulation results we further considered the DGD model to determine if any specific changes to it can result in better agreement with the experimental data. We tested several different such changes (Methods) based on either previously implied molecular behavior (for example, when do cells secrete the chemical attractant?) or changes that affect parameters that are used by the model (for example, tumbling frequency, message weight when integrating information, etc.). For each of the changes we re-run the simulations and compared their output to the experimental results. We identified two key changes needed for the model to better match the experimental results. The first affected the messages sent by each cell (secreted signaling molecules). While prior communication based models assume that cells secrete such signaling molecules continuously, we found that limiting the release of the attractant to only when a cell observes an improvement in the chemo-attractant gradient leads to better performance in the more complex environments. This change in the model is supported in part by a recent study where we observed that bacteria indeed secretes an attractant

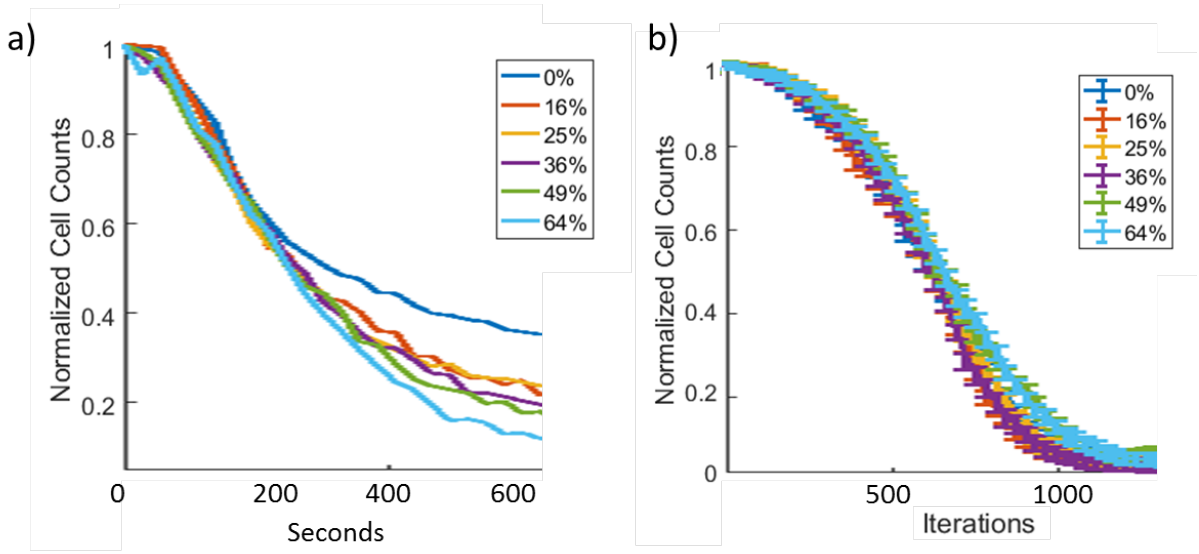

Figure 4.7: Experimental results and revised DGD model for cells in 200μM aspartate (Asp).(a) Total cell count of YFP-labeled, wild-type *E. coli* RP437 cells in the microchambers as a function of time, after applying 200μM Asp (in M9-G) as chemotaxis stimulus. Note the difference between the experimental results and simulation results from Fig. 1. (b) Simulations using the revised DGD model for the same settings (same as Fig. 4.5 c). By changing the secretion model and adjusting tumbling rates based on the environment, the DGD model leads to results that are concordant with the experimental results.

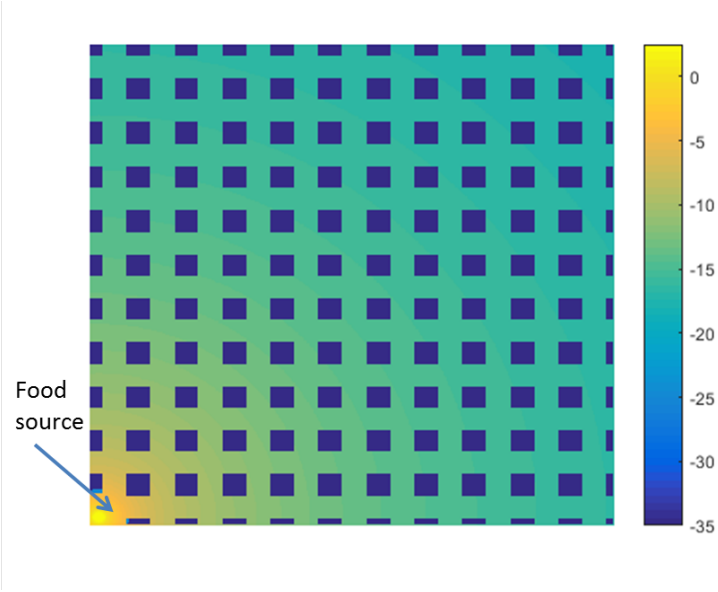

Figure 4.8: Gradient profile in simulation in the presence of obstacles. Gradient profile is roughly the same as when no obstacles are present (except, of course, to the location of the obstacles).

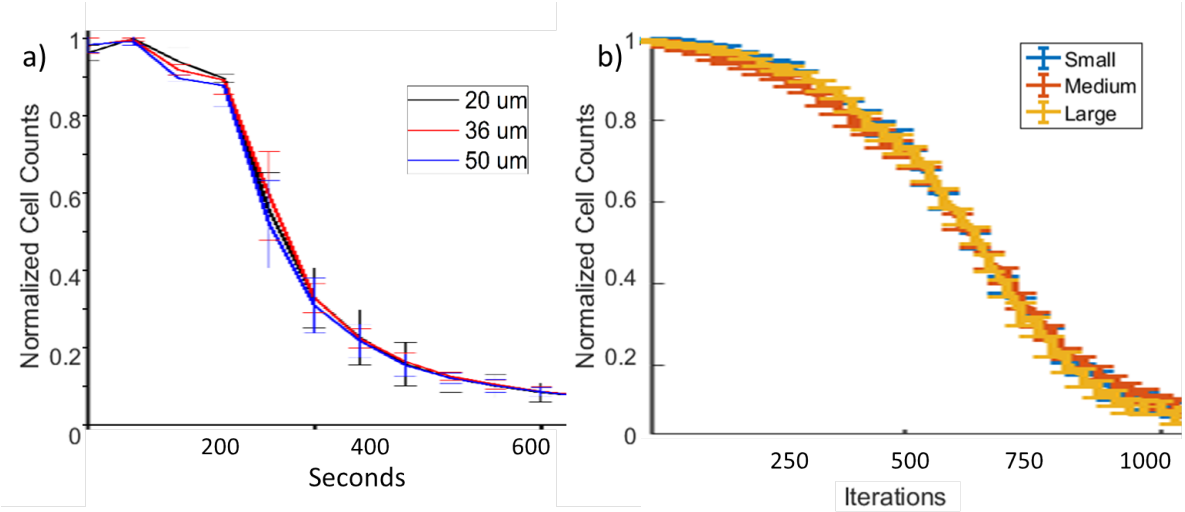

Figure 4.9: Performance of the revised DGD model in varying obstacle size. a) Experimental data: Normalized count of YFP-labeled, wild-type *E. coli* RP437 cells in the microchambers as a function of time, after applying a chemotaxis stimulus, M9CG. b) Simulation results from revised DGD model: Normalized cell count under different obstacle sizes.

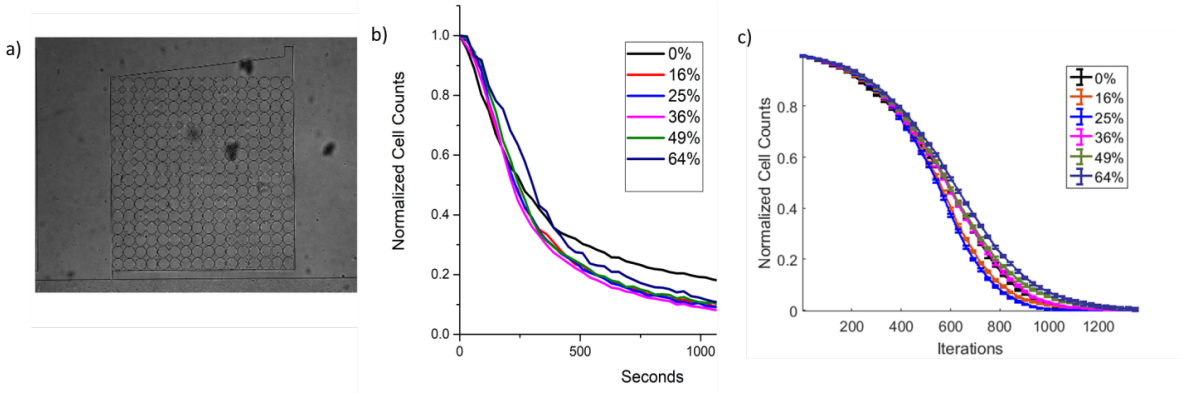

Figure 4.10: Experimental results and revised DGD model with round obstacles. We performed experiments with round obstacles to test whether the accuracy of the predictions by the revised model holds for different obstacle shape as well. (a) Design patterns of a microfluidic device with round obstacles. (b) Normalized cell count of YFP-labeled, wild-type *E. coli* RP437 cells in the microchambers as a function of time, after applying a M9CG chemotaxis stimulus. (c) Simulations using the revised DGD model for the same settings. Note that, the revised DGD model can accurately predict the experimental results for round shaped obstacles as well.

in response to sensing an external attractant gradient [20]. Note that while Shklarsh model also uses a confidence based binary weighting, our revised model uses a different formulation. The binary weighting in our model is based on sender confidence while the Shklarsh model is based on receiver confidence, meaning that the receiver can ignore messages from its neighbor when it assumes that it is moving towards the gradient [26].

The second and more surprising model change was related to a reduction in tumbling frequency based on the environment the cells occupy. We observed that reducing the tumbling frequency based on the number of times a cell is unable to proceed in a given direction due to obstacles in its path, leads to much faster exit times in the higher coverage environments (see Methods for mathematical and computational details, which are based on feedback loops, and how they impact individual cell behavior). With these changes, the simulation results of the updated DGD model exhibited qualitatively good agreement with the experimental results (Fig. 4.5c, 4.9). We also tested the impact of these changes on another prior model (Shklarsh model). However, even with these changes simulations based on the updated Shklarsh model did not agree with experimental results (Fig. 4.11).

4.5.4 Experimentally testing model predictions

The revised model leads to two predictions about the behavior of *E. coli* cells in complex environments. First, it predicts that tumbling is reduced when the obstacle coverage increases. Second, it predicts that such reduction is based on feedback, and thus over time, we expect a reduction in tumbling in cases where cells encounter an environment in which a large fraction of the surface is covered by obstacles. The first prediction is intuitive and stems from the limitations imposed on the cells movement by the physical obstacles. When a cell encounters a barrier, it moves parallel to it, which leads to a reduction in tumbling frequency [81]. However, the second prediction is surprising and we are not aware of prior work describing it. Such change implies that bacteria may have a long term adaptation mechanism.

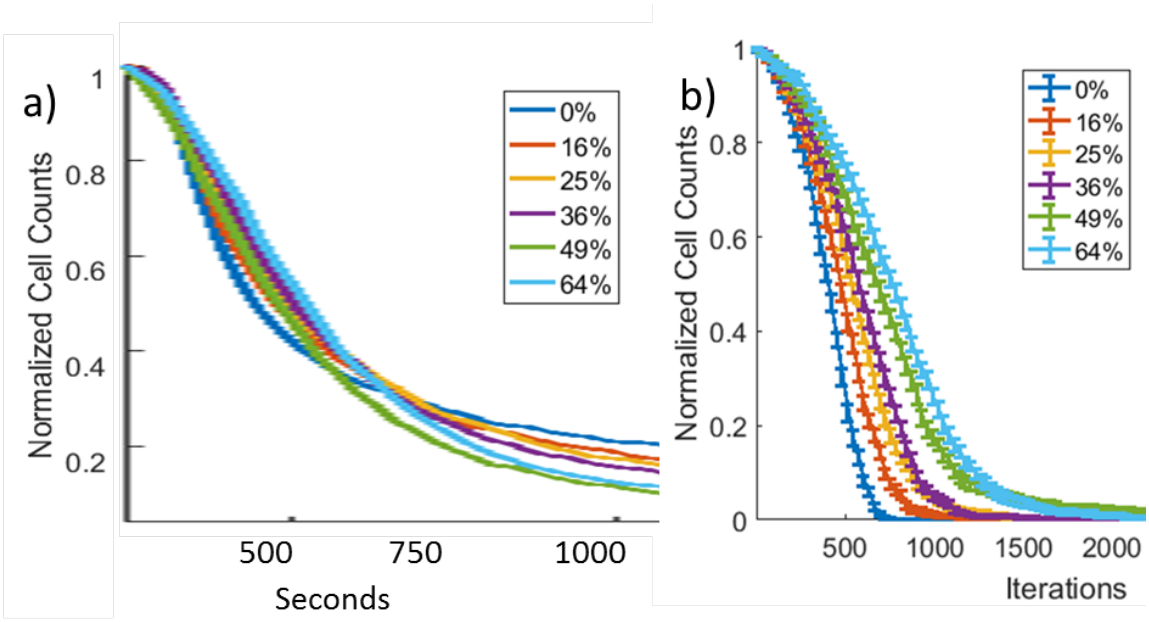

Figure 4.11: Performance of updated Shklarsh model. (a) Normalized cell count of YFP-labeled, wild-type *E. coli* RP437 cells in the microchambers as a function of time, after applying 200μM Asp (in M9-CG) as chemotaxis stimulus. (b) Simulations using the revised Shklarsh model for the same settings. As can be seen from the figure, even after updating the tumbling rate reduction, the Shklarsh model still does not agree with the observed outcomes for environments with and without obstacles. Specifically, using the same tumbling rate reduction as we have done for the DGD model the Shklarsh model still predicts a much faster time for no obstacles.

To test these predictions we analyzed the movement of individual cells in different environments by tracking their trajectories (Methods). The tracking algorithm links a cell in a frame with the closest cell in the next frame within a maximum distance, which was set to 10 pixels ($\sim 23\mu m$). Tracking was performed for three different terrains: An environment with no obstacles (0% coverage), with 64% square obstacle coverage, and with 64% round obstacle coverage. Fig. 4.12 displays a number of representative cell trajectories for these environments.

Given the trajectories of each cell, we next analyzed their tumbling frequencies, T_f . We define a 'tumble' when the angular movement of a cell is above a certain threshold (Methods). Since cells complete a tumble over a period of time, we based the tumble calls on multiple consecutive frames from the experimental movies. From the point of origin in the movie, or the end of the previous tumble, we have computed cumulative angular movement in the subsequent frames. When the cumulative angular movement reaches the pre-defined threshold, we call it a tumble and update the point of origin at the detected tumble location. This has been repeated until the tracking for the cell is completed (see Methods for complete details and algorithm). Detected tumbles for these environments are shown in Fig. 4.12 (bottom three rows). As can be seen, while the 64% coverage environments are more constrained, cells seem to perform fewer tumbles. In contrast, even though they are not facing any specific limit on their movement, in the unconstrained 0% environment cells seem to tumble more.

Fig. 4.13a,c quantifies these differences and presents the tumbling frequency distributions for the 0% and 64% environments. Tumbling frequency is computed by dividing the number of tumbles in each track by the duration of tracks in seconds. Distribution is calculated using tumbling frequencies for all the cell tracks. As can be seen, the average tumbling frequency is lower in the 64% coverage setup by $\sim 30\%$ when compared to the 0% coverage, as predicted by the revised model. Similarly, Fig. 4.13b,d compares the tumbling frequency for the earlier frames in the 64% environment to those in the later frames/times. Again, as predicted by the model we see a decrease ($\sim 10\%$) in the tumbling frequency over time indicating that cells can indeed

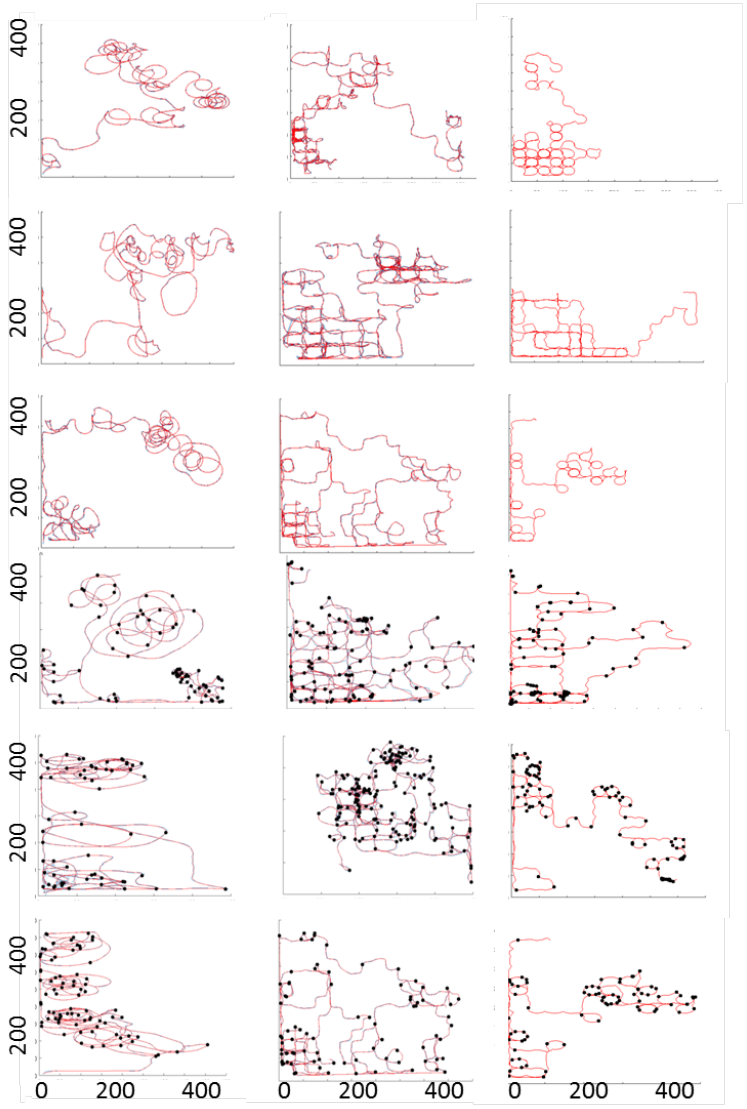

Figure 4.12: Single cell trajectories under different obstacle coverages. Left column (top 3 rows): Trajectories of cells when no obstacles are present. Middle column (top 3 rows): Trajectories with 64% of the area covered by square obstacles. Right column (top 3 rows): Trajectories with 64% of the area covered by round obstacles. Note, the grid like trajectories of individual cells are due to the obstacles. Bottom 3 rows: Additional trajectories without (Left column) and with (Middle column (square), Right column (round)) obstacles, this time with the identified tumbles marked by black dots. As discussed in the text and in Fig. 4, the analysis identifies more tumbles in the no obstacle cells even though they are much less constrained than the 64% group. The food source is located at the bottom left corner at (0,0).

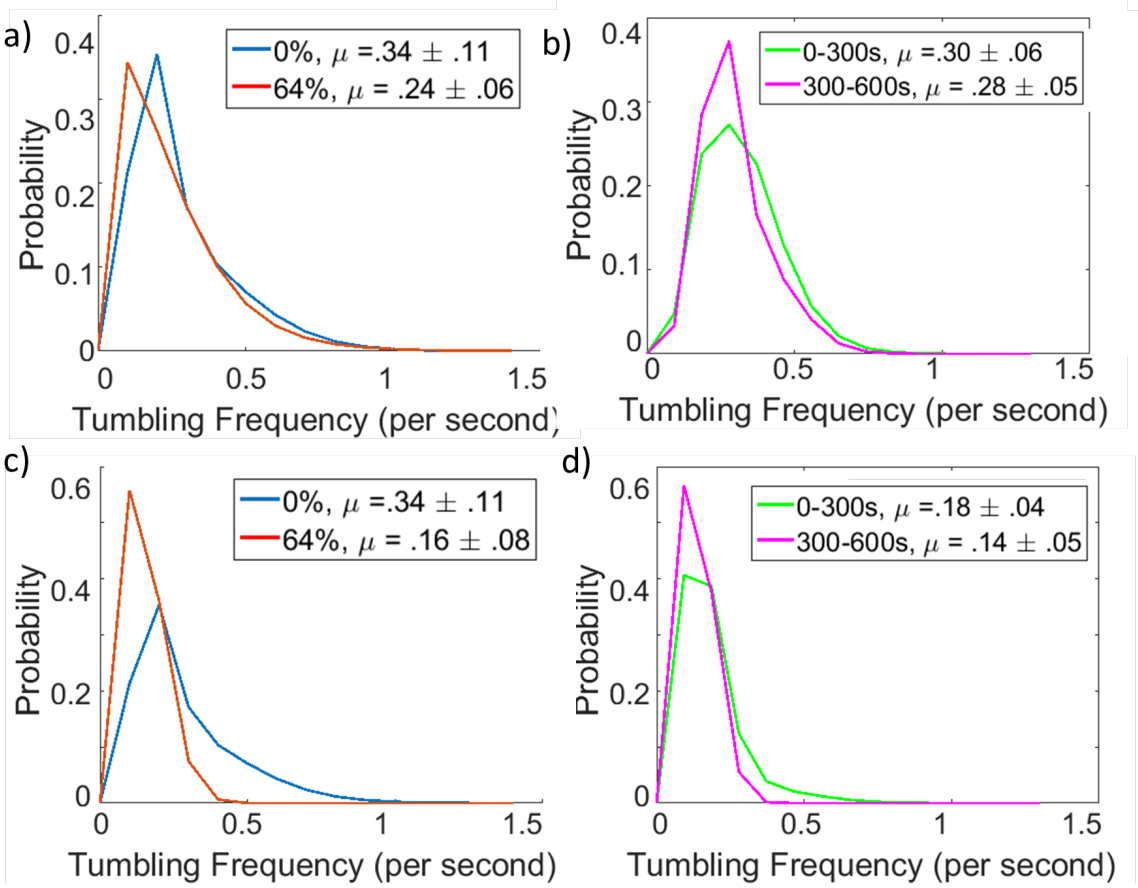

Figure 4.13: Distribution of tumbling frequency under different obstacle coverages. (a,c) Distribution and average of tumbling frequency (tumbles per second) for two different obstacle types: a) 0% (red) and 64% square obstacles (blue), c) 0% (red) and 64% round (blue). A shift in the mode to the left (lower) for the 64% case is evident, indicating that cells may indeed be learning to reduce their rates. (b,d) Distribution and average of tumbling frequency (tumbles per second) for cells in 64% (b) square, and (d) round, obstacle covered environments at two different times: early (first half of trajectory, green) and late times (second half, pink). Cells reduce their average tumbling frequency over time, especially if these rates are high to begin with (right hand side).

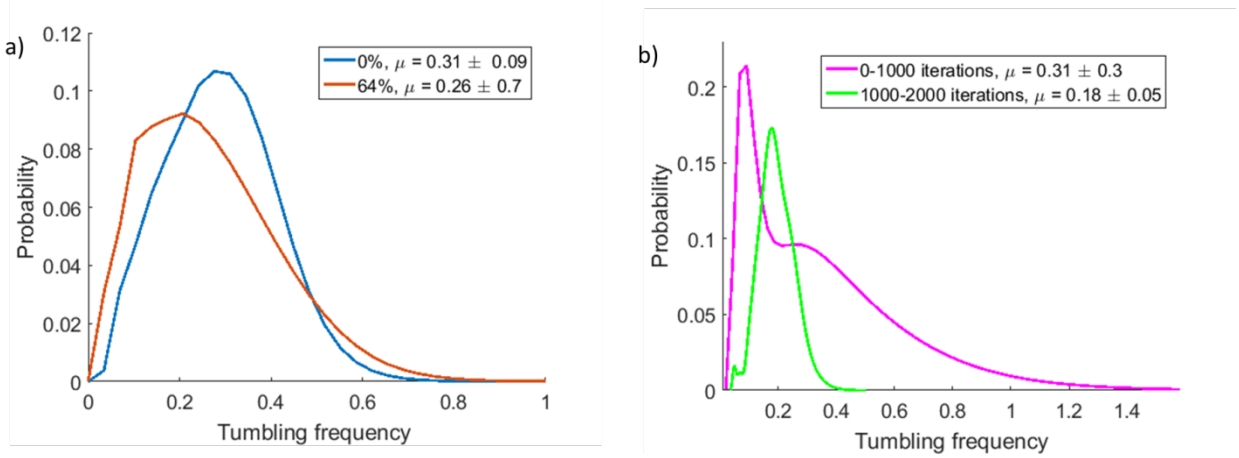

Figure 4.14: Distribution of tumbling frequency under different obstacle coverages in simulation. (a) Distribution and average of tumbling frequency (tumbles per second) for two different obstacle setups: 0% (red) and 64% (blue). A shift in the mode to the left (lower) for the 64% case is evident, indicating that cells may indeed be learning to reduce their rates. (b) Distribution and average of tumbling frequency (tumbles per second) for cells in 64% covered environments at two different times: early (first half of trajectory, pink) and late times (second half, green). Cells reduce their average tumbling frequency over time, especially if these rates are high to begin with (right hand side).

adjust their tumbling rates based on their environment. We observe similar tumbling distribution plots from simulation of the revised DGD model (Figs. 4.14, 4.15).

We have also performed global analysis using mean square displacement (MSD) to model tumbling in the single cell tracks. The MSD analysis results are presented in (Fig. 4.16), and are in good agreement with our previous findings. We find that t_c is larger in terrains with obstacles (i.e. lower tumbling frequency in the presence of obstacles), and that in the presence of obstacles it increases with time indicating that the tumbling frequency decreases in time. Note that this change in t_c or the tumbling frequency in time, cannot be attributed to the bacteria getting closer to the chemo-attractant source and therefore sensing a stronger gradient, since such effect will be observed in the microchambers without obstacles as well.

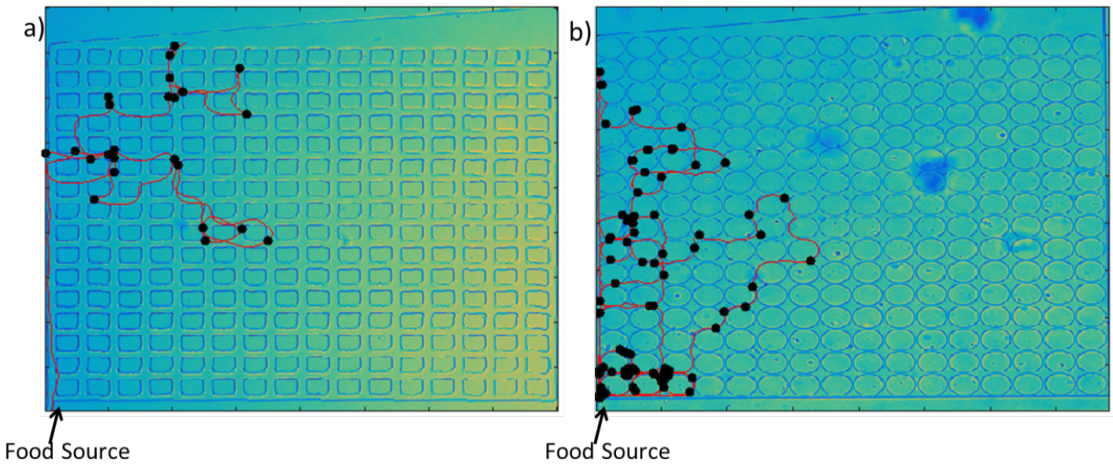

Figure 4.15: Cell tracks under 64% obstacle coverage simulations. a) Square obstacle, b) Round obstacle. Detected tumbles are marked by black dots.

In our setups all obstacles are of the same size and shape and are placed in a regular grid. To investigate whether the adaptive tumbling mechanism improves performance of bacterial chemotaxis for setups with irregular obstacles, we simulated both the original and revised DGD methods in a chamber with randomly placed obstacles (triangles, rectangles, pentagons, hexagons, and circles) of variable sizes (Fig. 4.17). For each setting we computed the number of iterations it takes for 90% of the cells to reach the food source in both models . Fig. 4.18 presents the distributions of time to reach the food source in three such irregular setups. The distributions were computed based on 50 independent trials. As can be seen, in each of the setups, the revised DGD model improves over the original DGD model.

4.5.5 Sensitivity analysis of the parameters of revised DGD model

There are two key changes in the updated model, i) the cells are now allowed to send messages only when they are moving in the correct direction (eq 4.8), ii) the cells increase their tumbling interval every time they hit an obstacle (eq 4.9). The first change is binary. Let $Q \in \{0, 1\}$ denote the change, $Q = 0$ for the original model and $Q = 1$ for the updated model. The other

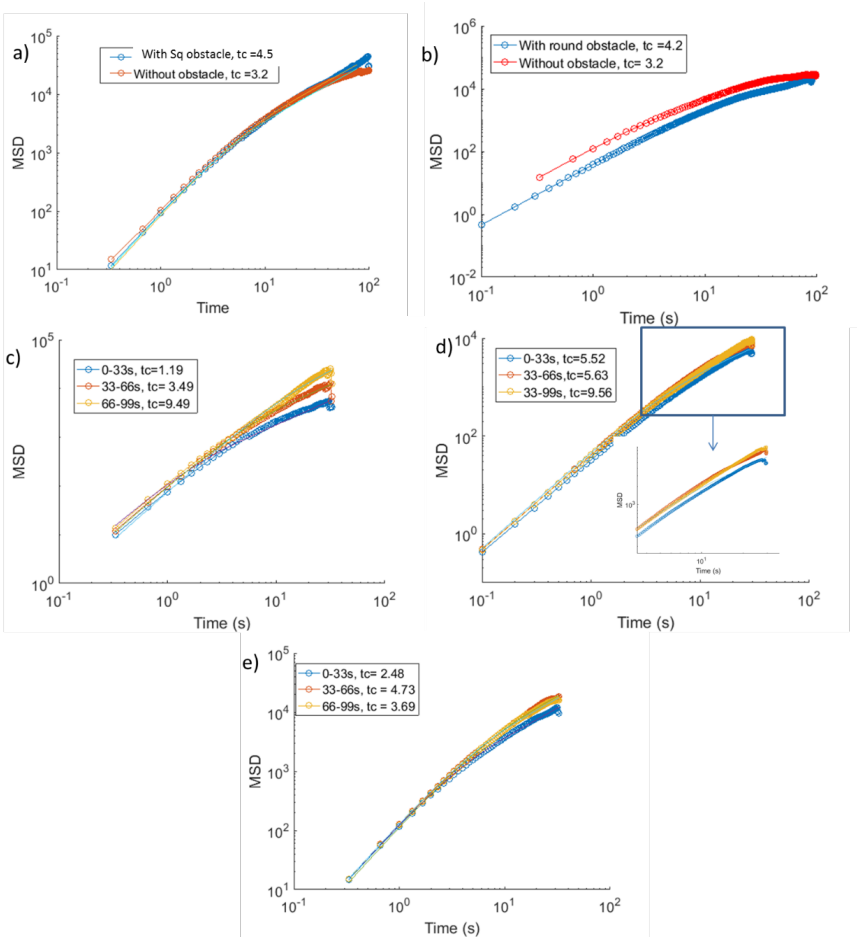

Figure 4.16: Log-log plot of MSD vs time (s) under different obstacle coverages. (a-b) MSD plot and critical time between tumbling (tc) for obstacle (a) square, b) round) and non-obstacle setup 0% (red) and 64% (blue). A shift in tc (higher) for both the 64% case is evident, indicating that cells may indeed be learning to reduce their rates. tc is critical time between tumbling at which the bacteria cell movement changes from ballistic to linear time scale. Even though for the round obstacles in b) the without obstacle plot is above the obstacle plot, the switch from ballistic to linear time scale happened much earlier compared to the obstacle setup. (c-d-e) MSD plot and critical time between tumbling (tc) over different time frames for the obstacle setup (c) square, d) round), and e). Cells reduce their average tumbling frequency over time for both cases, especially if these rates are high to begin with (right hand side). To better focus on the change of critical time between tumbling for round obstacles, we have zoomed the end part of the MSD plots (shown in inset).

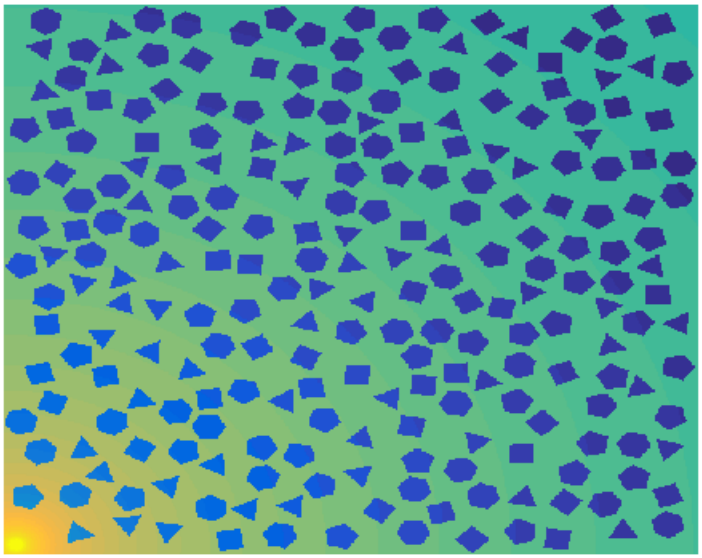

Figure 4.17: Irregular obstacle setup

change can be parameterized by f (eq 4.9). f is set at 0.0005 in our updated model. Setting the parameter to 0 will revert back to original model. Table 4.1 summarizes the sensitivity to the parameters. We have quantified the mean and variance of the required iterations for 90% of the cells to escape across all the obstacle coverage. We see the largest impact on the variance from Q (message sending) while the tumbling rate can have a high impact on the mean time (up to 20% difference). In our final model we use $f = 0.0005, Q = 1$. While the mean is slightly higher than the best value tested for f, variance is much lower and the fit to the experimental results the highest. We thus conclude that both parameters can have a large impact on performance and should be used by the model. In addition to the parameters of the revised DGD, Table 4.1 also shows sensitivity to the w paramter of the original DGD model. w is set at .5 in our model and changing the parameter within reasonable value ($[.3, .7]$) does not affect the performance too much.

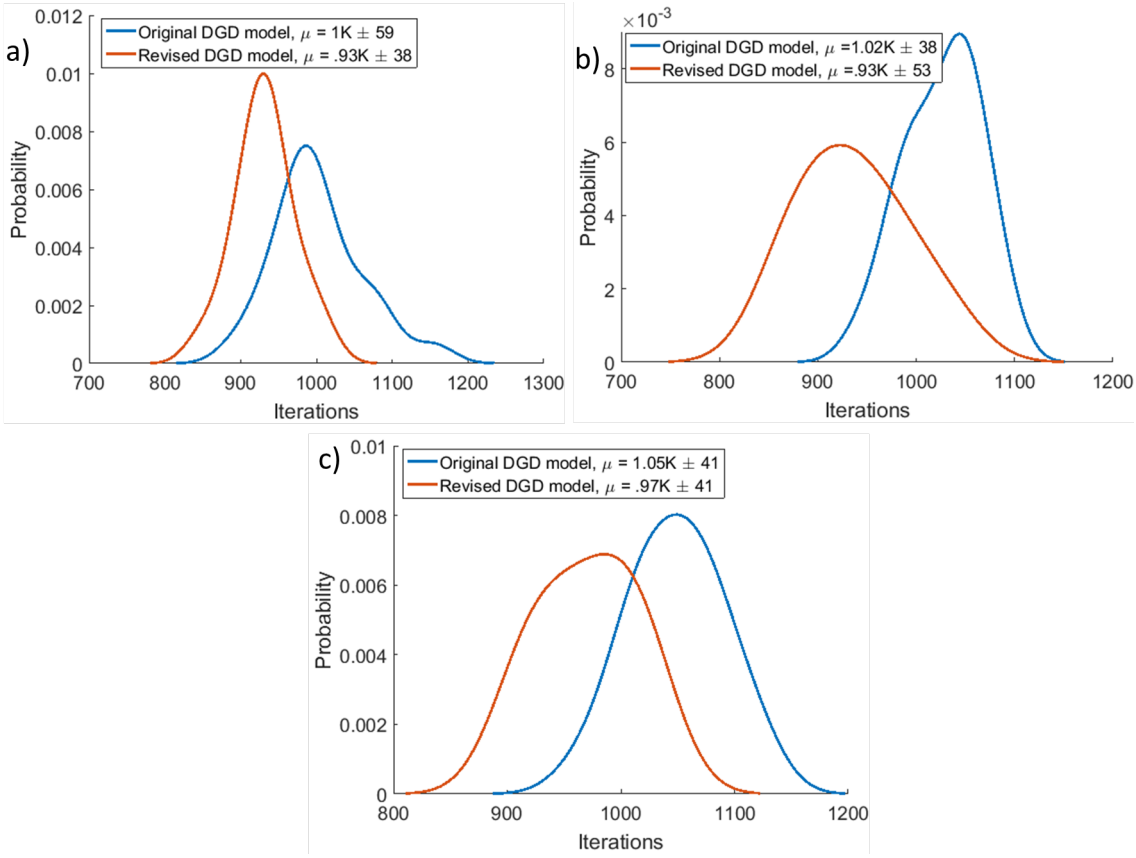

Figure 4.18: Distribution of bacterial chemotactic food search time for three random obstacle setups. Due to random shape and placement of obstacles the coverage varies between each random setup. The coverage in the three figures above is: a: 26%, b: 28% c: 32%. As can be seen, for all placements the revised DGD model performs better than the original DGD model. μ denotes the average time it takes the fastest 90% of the cells to reach the food source.

Q	f	w	Mean	Variance
1	0.0005	0.5	823	30
1	0.0005	0.3	815	32
1	0.0005	0.7	818	35
1	0.005	.5	720	90
1	0.005	.3	723	88
1	0.005	.7	729	104
1	0.00005	.5	910	50
1	0.00005	.3	903	49
1	0.00005	.7	916	52
0	0.0005	.5	967	417
0	0.0005	.3	962	411
0	0.0005	.7	969	434
0	0.005	.5	940	430
0	0.005	.3	942	421
0	0.005	.7	944	440
0	0.0005	.5	925	380
0	0.0005	.3	922	345
0	0.0005	.7	926	392

Table 4.1: Sensitivity analysis of revised DGD model parameters

4.5.6 Testing the revised DGD model on a computational system

Several distributed computing methods rely on swarm based approaches [27]. For example, methods for locating trapped victims in emergency situations are often based on robotic swarms [28, 73]. In such methods robots follow signals from victims (could be smell or sound) to determine their search route. In realistic scenario robots sensor reading of such signals are very noisy. In addition to that, the environment is often composed of many physical obstacles. To solve such complex problem setups, the robots in the swarm communicate with each other, often broadcasting the best local solution obtained by the individual robots to influence the search routes of the neighboring robots.

To test whether the idea of reduced tumbling in complex environments can be used in a swarm robot algorithm we used a simulation environment implemented by NetLogo [82]. In this setup, each robot is equipped with two sensors, one is capable of sensing sound / smell (to find victims which are the attractants here) and the other one locally senses neighboring agents or obstacles in the environment (visually, usually within a small fixed range). The setup of the NetLogo simulation is shown in Figure 4.19. The parameters of the NetLogo simulation is shown in Fig. 4.20. A recent benchmarking paper [83] identified the Darwinian Particle Swarm Optimization (DPSO) [84] as the most efficient in avoiding local minima in distributed search tasks. In DPSO, to simulate natural selection, many simultaneous, parallel PSO algorithms, each on a swarm, operate on a test problem. Simple rules are developed to implement selection. Similar to chemotaxis models, in DPSO, the agents continuously adjust their movement direction based on the sound / smell gradient from the victim and the perception of the obstacle. Each robot also influences the movement direction of other members of the swarm, by sending messages about the best local solution obtained by it. Combining this information with its own perception of gradient and environment, each robot decides on the movement direction in each iteration.

We tested DPSO in 50 independent runs with random setup of obstacles and noisy gradient for the task of finding victims. The results are presented in Figure 4.19. Next, we tried to

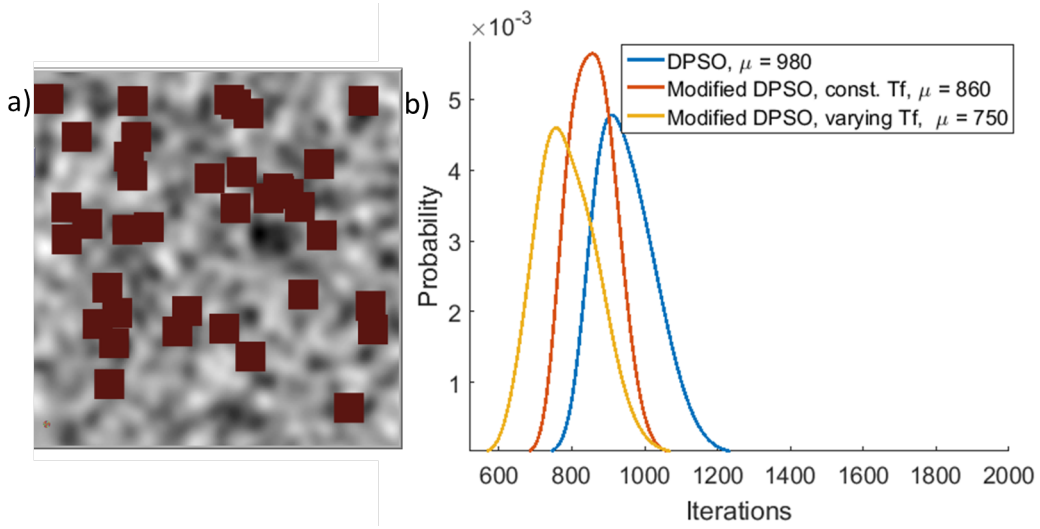

Figure 4.19: Application of adaptive tumbling frequency method for swarm robot search task. a) Terrain used in one of the NetLogo simulations. Grey shades correspond to attractant distribution. Red squares are randomly placed obstacles. b) Distribution of search time in DPSO and Modified DPSO with constant and varying tumbling frequency over 50 independent trials. By adjusting tumbling frequency based on the terrain the modified DSPO method improves search time.

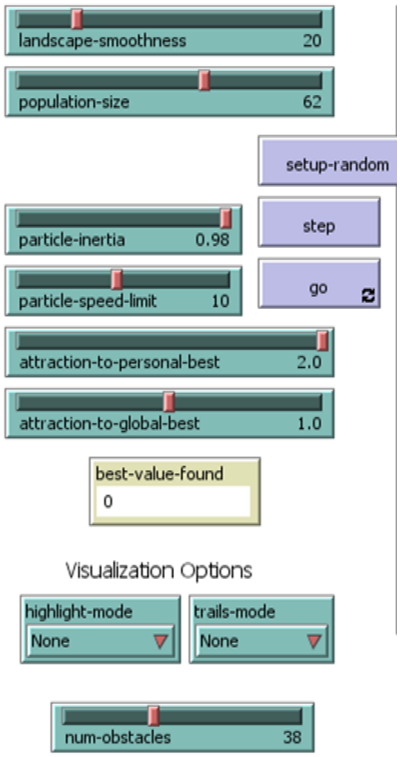

Figure 4.20: Simulation parameters used on NetLogo software.

determine if introduction of tumbling improves the performance of DPSO in search task. First, we forced each robot to tumble (change movement direction) in every few iterations (constant tumbling frequency). Given the noisy gradient, it is evident that the tumbling actually helps the robots to search more efficiently. Next, we tested adaptive tumbling mechanism, here the interval between two consecutive tumbles are determined adaptively based on how many times the robot encounters an obstacle. Tumbling frequency actually decreases the more the robots encounter an obstacle. It decreases the probability of robots repeatedly encountering obstacles, hence improving the search performance.

4.6 Discussion

Several studies have focused on organism and cell movement. For example, in [85] the authors model ants movement and in [86] the authors model zooplanktons movement response to light source. Other studies have attempted to model movement of bacteria in response to a chemical stimulus in different chambers and for several different attractants [87, 88, 89]. However, as we showed these models cannot accurately predict bacterial behavior in complex environments. Bacterial cells face many obstacles in their natural environments to reach their target. Whether it is a nutrient source, or host cell for pathogens, bacteria often have to navigate through confounded terrains. *E. coli*, for example, naturally inhabits the gastrointestinal tract, which is a complex environment and can impose complicated topology with physical obstacles that would hinder the progress of the bacteria towards their target. Therefore, it is likely that their behavior in such environments has evolved to minimize their time to reach an attractant nutrient source (e.g., aspartate). To uncover likely mechanisms used by bacteria to reduce search time in such complex environments we considered several possible changes to mathematical models of such behavior that can lead to observed exit times of cells. We found that in addition to limiting the secretion of cell-cell signaling to times when the cell is moving up a gradient, a key change that greatly improves exit times is reducing tumbling rates based on the number of times the cells

encounter a physical obstacle. In other words, cells seem to learn (form short term memory of) the complexity of the environments they are in and adjust their tumbling rates accordingly. While the control of such learning mechanism is not known, adaptation to the environment is very common in bacteria. For example, bacteria adapt their sensing ability to their surroundings as they move up an attractant (or down a repellent) gradient, such that they are always able to sense a relative improvement in the conditions [90]. In this case, it is possible that as the cell is prevented from tumbling by the physical barriers it encounters, it reduces its tumbling frequency. Such a feedback would be controlled through the cells mechano-sensing apparatus, whose effects has been reported, but its control mechanism is yet to be discovered [91, 92].

Applying the changes mentioned above, leads to models that mimic observed behavior under a wide range of obstacle coverages. While we still observe some differences between experiments and simulations, these are likely the result of cells that get stuck to the plate, or due to changes in the external gradient over time. While most cells manage to escape before there is a significant change in the gradient, at some point the microchamber becomes saturated with the external signal preventing remaining cells from escaping (for more details of the gradient change over time see [20]). However, this difference does not impact the conclusions of our modeling and experimental results which indicate that the decrease in tumbling rate is the likely reason for the faster than expected) exit of most cells (Fig. 4.4).

To test if the model predictions indeed reflect bacterial behavior we have tracked single cells in settings with and without obstacles. These experiments revealed that when faced with terrains that are covered by a large number of obstacles (64% total surface coverage) cells indeed reduce their overall tumbling rates when compared to those observed without any obstacles (0% coverage). Further, we have shown that this behavior is indeed adaptive. Even for the 64% coverage case we observe a reduction of tumbling rates over time with higher initial tumbling, which is reduced as the cells encounter more and more obstacles. We note, however, that in our analysis, we do not distinguish between tumbling due to intracellular mechanisms or due to the

physical environment. This is because both would have the same effect on the overall motion of the bacteria. However, the fact that there is a change in the tumbling frequency over time in the same terrain indicates that the intracellular control mechanism is influenced by the physical environment, which does not change over time. Future studies that aim to identify the molecular mechanism by which *E.coli* cells control such rapid adjustment in their tumbling rates, are still needed.

An interesting question that this study raises is how the migration of bacteria away from repellents would be influenced by physical obstacles. The consensus in the field is that the bacterial migration away from repellents is driven by the same mechanism as its migration towards attractants. However, an important distinction should be made here. Our results, as well as previous studies [20] point to a cell-cell communication mechanism in which secretion of the attractant is triggered by the sensing of the external signal. This might be similar when bacteria are moving down a repellent gradient if the secretion is triggered by any improvement in the environment. However, if it is triggered by the sensing of a specific chemical, then one would expect that the migration of bacteria down a repellent gradient will differ significantly from the pattern observed here. This question remains open for future studies.

Reasoning under uncertainty, in which a collection of agents needs to reach a common goal while each has access to limited information is also of major interest in the Artificial Intelligence (AI) community. Current algorithms for this problem utilize robotics swarms that often operate in noisy and complex environments [73]. However, most AI algorithms for this task do not allow feedback from the environment or topology to adjust the hyper-parameters used in the algorithm. To test whether the idea of reduced tumbling in complex environments can improve swarm based searches we used a simulation environment implemented by NetLogo [82] (Methods). A recent benchmark paper [83] identified the Darwinian Particle Swarm Optimization (DPSO) [84] as the most efficient current method in avoiding local minima in these distributed search tasks. We compared three possible search methods: No tumbling, fixed tumbling, and adaptive tumbling

similar to the behavior observed for bacterial cells. As can be seen, we observe an improvement (reduction in search times) when using adaptive tumbling in noisy environments. These results show, that adopting insights based on bacterial coordination can suggest new directions to improve the performance of these algorithms. More generally, our results provide additional support to the usefulness of studies that attempt to determine what and how biological processes compute, and in turn use the results to improve computational methods [79, 80].

Chapter 5

Genome Wide Epigenetic Modifications as a Shared Memory Consensus Problem

5.1 Introduction

While message passing (e.g. in bacterial DGD) is a useful and dominant method for distributed biological computing, some biological processes utilize another method for coordination. This method is similar to the method of communicating via shared memory, studied intensively in distributed computing, and utilizes modification to the DNA which can be sensed (read) by other participating entities. This process is termed *epigenetics* and involves several different groups of proteins (histone modifiers) which could largely be divided into three groups: *Readers, Writers, and Erasers* [42].

Although a lot of recent work in the biological literature has focused on histone modifications and the specific proteins that regulate this process, to the best of our knowledge no prior work discussed the issue of global coordination between writers and erasers. We often see *exactly the same* histone marks for a stretch of DNA [41]. However, how such consensus is reached for a particular mark is still not clear. Our goal in this paper is to understand better how such consensus is reached.

To this end, we model the histone modifiers as two different types of writer processors and two different types of eraser processors that communicate by accessing a shared memory array (a stretch of DNA), and for such a setting formally define the *consensus write-erase problem*. We first discuss a simple algorithm for solving the problem and then present a more sophisticated algorithm, that better matches various biological assumptions, and discuss its run time both theoretically and in simulations.

5.2 Model

We assume a shared memory array with N cells, which corresponds to a linear DNA stretch with N total histones that can be modified. Recent biological results indicate that such stretches of DNA, often anchored by CTCF bindings sites, are likely to be jointly modified when switching between cell states [93]. Thus, our focus here is on achieving a consensus assignment for such stretches when cells need to switch between different biological states.

Since our processors mimic memory-less proteins, we further assume that each processor has only two bits of memory which, for example, prevents it from counting. While there could be several types of modifications, many of them can co-exist (each changing a different histone residue). For a specific residue most of the modification are restricted to two possible values and so we assume here that only two values can be written to each location, denoted by 0 and 1. Again, following the biological model we allow each processor to be either a writer or an eraser [94].

We assume a set of 0-writers and 1-writers is assigned to each DNA stretch. W_0 denotes the number of 0-writers and W_1 denotes the number of 1-writers. Similarly, we have sets of erasers for 0 and for 1, such that 0-erasers can only erase 0 and 1-erasers can only erase 1, respectfully. We assume that the generation of writers and erasers is similarly regulated and so number of 0-erasers, $E_0 = W_1$ and number of 1-erasers, $E_1 = W_0$.

To switch between states, cells transcribe (generate) new writers and erasers for the mod-

ification needed (for example, when changing from 0 to 1). When changing from 0 to 1, the new 1-writers will usually outnumber the existing 0-writers. However, we cannot expect these 0-writers to completely disappear, at least not within the time scales needed for changing the state. Thus, we assume that at any given time the number of one type of writers is larger than the number of the other type, and the convergence time of our algorithms will depend on this assumption.

Each of the N locations in the shared memory can be in one of three states: Empty (V), 0 or 1. A state transition can occur from V to 0 (1) by a 0 (1) writer and from 0 (1) to V by a 0 (1) eraser. However, a transition from 0 (1) to 1 (0) cannot occur. That is, a 0 state needs to be erased first and only then can be written by a 1-writer. It is assumed that reading a cell and then possibly updating it is done as one atomic step. In addition, it is assumed that all locations are initially empty.

We assume that writers and erasers are *asynchronous*, nothing is assumed about their relative speed. The time efficiency of an asynchronous algorithm is often of crucial importance. However, it is difficult to find the appropriate definition of time complexity, for systems where nothing is assumed about the speed of the processors. Thus, for measuring time, we will assume that a single step of a processor takes at most one time unit.

Assume that a computation is taking place through time and that every step of every processor takes some amount in the interval (0,1]. That is, there is an upper bound 1 for step time but no lower bound. Thus, for example, if during some period of time, where two processors are taking steps, one processor takes 100 steps while the other takes 5 steps, then the total time that has elapsed is at most 5 time units.

Under the assumption that a single step of a processor takes at most one time unit, the time complexity of an algorithm is defined as the maximum number of time units (also called "big steps") that are required for the algorithm to converge (i.e., to terminate) [95, 96, 97]. For the rest of the paper, by a step of the algorithm, we mean a "big step" which takes one time unit

and where each correct processor that started participating in the algorithm has taken at least one step. For example, in Section 5.4 we derive the expected number of big steps (i.e., time units) for our consensus algorithm to converge.

5.3 The consensus write-erase problem

The *consensus write-erase problem* is to design an algorithm in which all processors reach agreement based on their initial opinions. In our context, reaching agreement is expressed by guaranteeing a consensus outcome of either 0 or 1 for all N cells. A consensus write-erase algorithm is an algorithm that produces such an agreement, assuming only writers and erasers as defined previously.

More formally the problem is defined as follows. There are a fixed number of W_0 0-writers, W_1 1-writers, E_0 0-erasers, and E_1 0-erasers. Initially, each one of the N cells is empty, and upon termination, the value of each cell is either 0 or 1. The requirements of the consensus write-erase problem are that there exist a *decision value* $v \in \{0, 1\}$ such that,

- *Agreement*: With probability 1, the value of each one of the N cells is eventually v, and does not change thereafter.
- *Validity*: There is at least one v-writer.

We point out that the first requirement has two parts. This first (the agreement part) is that all cells eventually contain the same value, and the second (the termination part) is that eventually the cells do not change there agreed upon value. The second requirement ensures that if $W_0 = 0$ then the decision can not be on the value 0, and similarly if $W_1 = 0$ then the decision can not be on the value 1. Thus, it precludes a solution which always decides 1 (resp. 0). The consensus problem defined above is also called *binary* consensus as the decision value v is either 0 or 1. It is required that a solution to the problem be *symmetric*, that is, the solution should not favor one of the two possible decision values. A generalization of the problem where the v is taken from a

larger set is not considered in this paper.

The consensus problem is a fundamental coordination problem and is at the core of many algorithms for distributed applications. The problem was formally presented in [98, 99]. Many (deterministic and randomized) consensus algorithms have been proposed for shared memory systems. Few examples are [45, 100, 101, 102, 103, 104, 105]. Dozens of papers have been published on solving the consensus problem in various messages passing models. Few examples are [106, 107, 108, 109, 110]. For a survey on asynchronous randomized consensus algorithms see [111].

5.4 Algorithms and analysis

5.4.1 A naive algorithm

Before we present our main algorithm, we first discuss a straightforward but non-desirable solution which is easy to analyze. In this solution, the erasers do not participate. Writers compete on writing the leftmost cell and the value written into that cell becomes the final agreed upon value. This is done as follows. Assume v is the written value. The v-writers continue writing v into all the cells. The major downside of this solution is that the probability of ending with the majority value is $p = W_1/(W_0 + W_1)$ (assuming a 1 majority) which is usually very dangerous for cells since there is a constant probability of not reaching the desired state. Furthermore, this solution assumes that all the writers have the same orientation (i.e., they a priori agree on which side is the left side) which is an unacceptable assumption. We present below a much better solution.

5.4.2 The consensus write-erase algorithm

While the algorithm above will finish in N steps, as mentioned it may not lead to the desired outcome. Instead, we propose to rely on recent biological observations that indicate that stretches of consecutive 1's (resp. 0's) are locally extended until they reach other stretches of 1's (resp. 0's).

Based on this we propose the following algorithm. Let $v \in \{0, 1\}$.

- Each of the writers and erasers starts at a random location. Their direction is also chosen randomly.
- *Rule for a v-writer*: The writer starts moving in the chosen direction. If it sees an empty cell, it writes v and moves on to the next cell. When a v-writer reaches the end of the stretch, it reverses its direction and continues. [1]
- *Rule for a v-erasers*: The v-eraser starts moving in the chosen direction. When it sees a value v which is preceded by the value $1 - v$, it erases the v. Otherwise, it just moves on. When a v-eraser reaches the end of the stretch, it reverses its direction and continues.

When the values of two consecutive non-empty cells are different, we call that a *collision*. The algorithm will run until all the cells' values are non-empty and until all collisions are resolved. Intuitively, each time there is an empty cell after a collision is resolved, the probability that the value 1 will be written is higher (assuming a 1 majority), and thus the algorithm will eventually converge.

5.4.3 Reaching consensus for biased coin flips

We take a brief detour to present an abstract coin flipping consensus problem. As we will discuss below, the analysis of the runtime for this problem is directly related to the analysis of the runtime of our consensus write-erase problem.

Consider a set of N coins in a row each with probability $p > 0.5$ for heads. We initially flip all coins. Next, we chose at random a pair of neighboring coins where one is heads and the other is tails (such a situation is called a collision) and with probability p (resp. $1 - p$) pick the tails (resp. heads) coin and flip it. We continue until all coins show the same value. See Fig. 5.1 for an example.

[1] Another version of the algorithm we considered, forces a writer to spin (wait) when it notices a collision. Simulations indicate that the spinning version is more efficient than the non-spinning one.

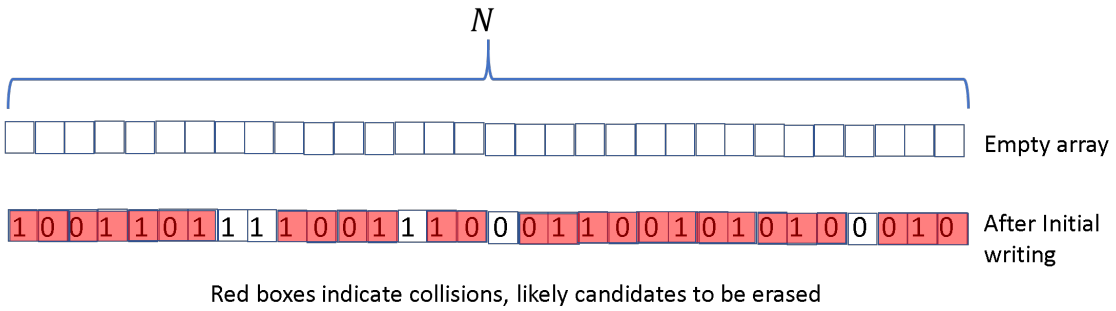

Figure 5.1: Biased coin flipping (1 - heads, 0 - tails). Red boxes indicate location cells with collisions where two neighboring coins show different values. In each round we pick at random one of these pairs and flip one of the two coins (with probability p for selecting the tails coin) until all coins show the same value.

5.4.4 Runtime analysis for biased coin flipping

Note that each step has a probability of p^2 (resp. $(1-p)^2$) to resolve a collision so that both coins become heads (resp. tails). Let $P = \frac{p^2}{(p^2+(1-p)^2)}$, and $Q = 1 - P$. Then conditioned on some change happening, P is the probability that the number of heads increases, and Q is the probability it decreases. Note that at each step where the count changes, the difference between heads and tails (number of heads - number of tails) increases by 2 with probability P, and decreases by 2 with probability Q. Since $p > 0.5$ there is a constant expected drift of $2(P - Q)$ in the heads direction.

Initially, the expected difference between the number of heads and tails is, $pN - (1-p)N = (2p-1)N$. The final difference after reaching consensus is N. Hence, to achieve consensus we have to close a gap (final difference-initial difference) of $N - (2p-1)N = 2(1-p)N$. Let T_{coin} be the number of steps needed to reach consensus. Then given the expected gap of $2(1-p)N$ the expected T_{coin} is

$$E[T_{coin}] = \frac{2(1-p)N}{2(P-Q)} \frac{1}{(p^2+(1-p)^2)} \quad (5.1)$$

$$E[T_{coin}] = \frac{(1-p)N}{(2p-1)} \quad (5.2)$$

Note that we multiply by $\frac{1}{(p^2+(1-p)^2)}$ to account for our condition that a change happens in this step.[2]

5.4.5 Using the analysis for the coin-flip consensus to analyze the write-erase consensus

Similar to the coin flip consensus problem, our write-erase consensus algorithm focused on neighboring cells with different marks (10 or 01, which we term as collisions). Since the algorithm is asynchronous and initial locations are assumed to be random, for each collision we have a probability $p^2 = (\frac{W_1}{W_1+W_0})^2$ of turning to 11 and $(1-p)^2 = (\frac{W0}{W1+W0})^2$ of turning to 00. Thus, the same number of flips is required to reach consensus. However, the number of steps needed to do such flips is different in our consensus write erase problem. In the coin flip problem we assume that at each step one colliding pair is flipped. In contrast, the time it takes to perform a flip in the write-erase algorithm is longer since we need the erasers and writers to arrive at the collision location. The expected number of steps it takes to perform a flip (T_{flip}) (i.e. erase and write) at a specific location is:

$$E[T_{flip}] = \frac{W_1^2}{(W_1+W_0)^2} \frac{2N}{W_1} + \frac{W_0^2}{(W_1+W_0)^2} \frac{2N}{W_0} + 2\frac{W_1W_0}{(W_1+W_0)^2}(\frac{N}{W_1} + \frac{N}{W_0}) \quad (5.3)$$

$$E[T_{flip}] = \frac{4N}{W_1+W_0} \quad (5.4)$$

The first part computes the expected time for changing a 0 to a 1 and the probability of such flip. The second computes the expected time to change a 1 to a 0 and the third is the probability of no

[2]When considering a *synchronous* model, in which all the coins which are involved in a collision are flipped again together in one step (round), it seems that the expected number of steps needed to reach consensus is logarithmic in N (rather than linear).

change (erasing 0 and writing 0 or erasing 1 and writing 1).

Given this, we can compute the expected time to solve the write-erase consensus problem. Let T be the number of steps to reach consensus, then:

$$E[T] = E[T_{coin}]E[T_{flip}] \tag{5.5}$$

$$E[T] = \frac{(1-p)N}{2p-1}\frac{4N}{W_1+W_0} \tag{5.6}$$

$$E[T] = \frac{4(1-p)N^2}{(2p-1)(W_1+W_0)} \tag{5.7}$$

Replacing p with $\frac{W_1}{W_1+W_0}$, we get

$$E[T] = \frac{4W_0N^2}{W_1^2 - W_0^2} \tag{5.8}$$

5.4.6 Resolving multiple collisions in parallel

The above analysis leads to a runtime of $O(N^2)$. However, the above analysis assumes a single collision being resolved each time. In practice, we have multiple collisions and so flips can co-occur. Initially, the number of collisions is a linear function of the number of 0's (since, for each 0 we have a probability $1-(1-p)^2$ that a 1 is written on either side). If the number of collisions remains a linear function of the number of 0's then in $O(N)$ steps we would flip $O(N)$ times (the initial number of 0's is $O(N)$) which would lead to at most $O(N \lg N)$ runtime since initially we have $O(N)$ 0's so the initial $O(N)$ flips will be in $O(N)$ time. Unfortunately, as Fig. 5.3a) shows based on simulations, this is not the case. The write-erase consensus algorithm leads to a rapid decrease in the number of collisions while not decreasing the number of 0's at the same rate. This means that long stretches of 0's (and 1's) form leading to a small number of collisions while still having a large number of 0's.

One way to overcome this is to change the algorithm to better mimic what biology does. Specifically, in biology we observe that erasers and writers interact during the establishment of a new state [112]. We hypothesize that an algorithm that utilizes these ideas can indeed lead to a faster runtime as the simulation analysis below shows, but at the present we are unable to prove

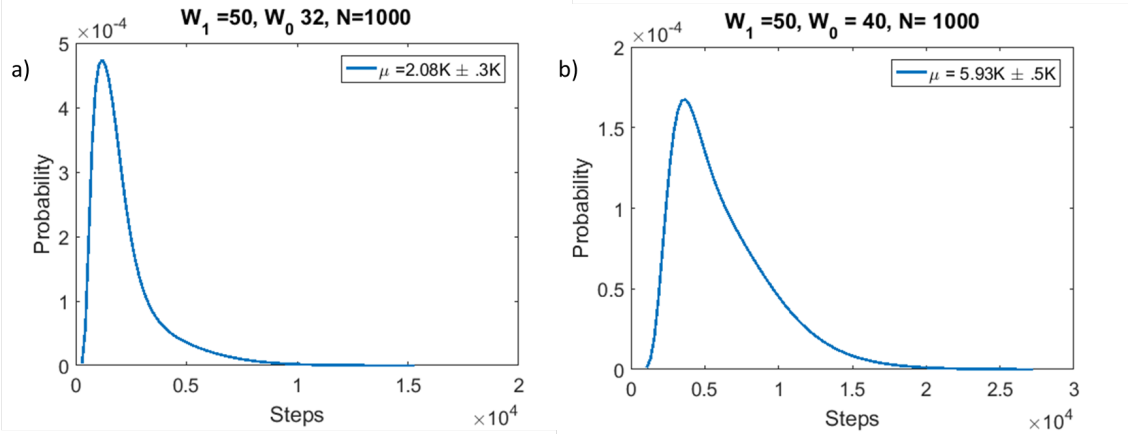

Figure 5.2: Distribution of number of steps to reach consensus. Plots summarize 300 random runs of the algorithm.a) low and b)high level of competitions between 1-writers and 0-writers. μ denotes the average time to reach consensus.

such a claim.

5.5 Simulations and analysis of real biological data

We performed simulations of our proposed algorithm at two different levels of competitions between the 1-writers and 0-writers. Fig. 5.2 shows the probability distribution of the number of steps to reach consensus. To simulate high level of competition we used $W_1 = 50, W_0 = 40, p = 0.56$. To simulate low level of competition we used $W_1 = 50, W_0 = 32, p = 0.60$. For both cases, $N = 1000$. As expected, at low level of competition consensus is achieved much faster. The average number of steps taken to reach consensus is $2.08k$ compared to $5.93k$ for the high level of competition.

Fig. 5.3(a) shows the change in the number of zeros and collisions as the algorithm progresses towards consensus. As can be seen from the figure, initially the number of collisions is linear in the number of 0's. However, the algorithm proposed leads to a quick drop in the number of collisions while the number of 0's remains fairly high which means that collisions cannot be resolved in parallel. We also simulated an alternative, which allows 1-writers to attach to 0-erasers

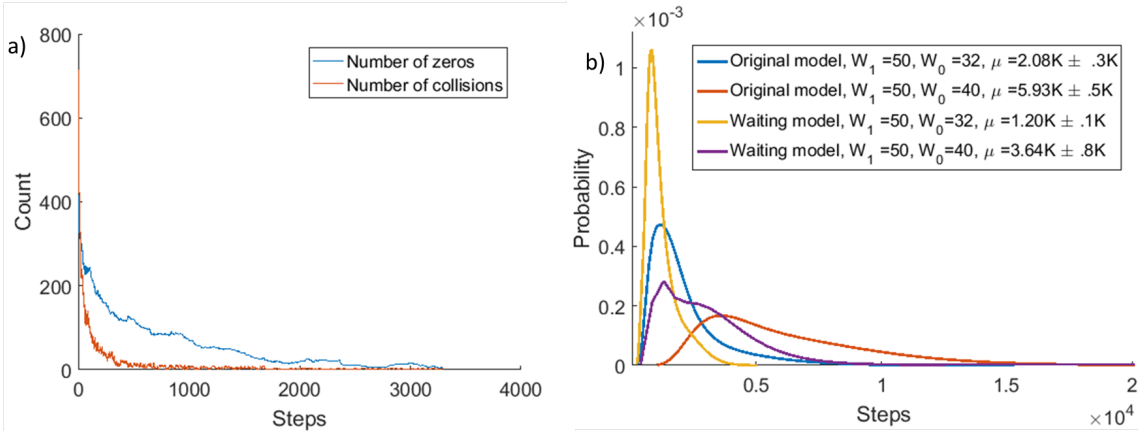

Figure 5.3: a) Number of zeros and collisions vs steps in the algorithm. While the initial number of collisions is a linear function of the number of 0's, we observe that towards the end of the algorithm there are very few collisions while the number of 0's remains relatively high. b) Comparison between the proposed original model and a revised model that allows writers to attach themselves with the erasers and the erasers wait until a collision is resolved. Here we can see that the waiting version is faster at both competition level compared to the original model.

(forming a writer-eraser complex though not guaranteeing atomicity). As before, erasers continue to scan the DNA until they reach a collision. However, in the revised version erasers wait at the collision site to see what value was written and if the new value leads to another collision they attempt to erase again until the collision is resolved. As mentioned above, this method is deadlock free if the ratio of erasers from the two types is not 1 and so would converge to a consensus. While we are unable to prove a better worst case runtime for such a method, simulations indicate that it leads to much faster convergence when compared to the consensus write-erase algorithm (Fig. 5.3b). Coupled with recent biological observations [113] these results indicate that this method is likely much faster than our current proposed algorithm while still not requiring any memory for the processors. We have also looked at recent epigenetic data to see if the assumptions we made about increasing stretches using collisions rather than randomly changing location are observed in real data. Fig. 5.4 presents results from a recent study by Gunther *et al* [41] in which two different types of marks, acetylation and methylation are competing for the same residue H3K9 (and so can be thought of as 0 and 1). As can be seen, and in agreement with our local consensus formulation, there is regional consensus of these marks. The methylation mark is strong in the region that lies between 30K bps to 60K bps and again from 105k bps to 120K bps. The acetylation marks are almost non existent in these regions, demonstrating regional consensus of methylation marks in the DNA and the likely impact of collisions on the establishement of such regions .

These consensus states are dynamically regulated in response to stress or for establishing a new state during development. Fig. 5.5 shows results from a temporal study by Gunther *et al* [41] in which cells are gradually moving towards consensus of tri-methylation marks of Histone 3 Lysine 4 (H3K4) residue as a response to Kaposi Sarcoma infection. The blue lines indicate presence of the H3K4me3 mark. The figure presents two time points, the first is at onset of infection (when the virus is applied) while the second one is from a sample 5 days post infection. As can be seen, after 5 days we observe a higher level of consensus than at the onset of infection

indicating that collisions continue to be resolved until full consensus is reached. Most likely, in this case, the reason for the increase is the activation of additional modifiers at the later time points which lead to changes in the ratio of 1-writers and 0-writers and erasers. We note that the studies performed to-date are looking at a collection at cells at once and so only report average data. New technologies are enabling us, for the first time, to observe these events at a single cell resolution [114] and we expect that these results will further help us infer the specific algorithms utilized by cell to reach consensus.

5.6 Discussion

To date, the study of "algorithms in nature" at the molecular and cellular levels, i.e., how collections of molecules and cells process information and solve computational problems, was discussed mainly in the context networks and message passing [2]. To the best of our knowledge, this thesis is the first attempt to study biologically inspired distributed computing algorithms in the context of a molecular shared memory systems.

We have focused on the process of genome wide epigenetic modifications in which cells utilize DNA as a shared memory object to establish a new state. We formulated a consensus problem that these modifiers need to solve and presented algorithms and their expected run time. We have also discussed and simulated improved methods for solving the problem which rely on additional recent biological insights. By analyzing real biological data we show that the decisions made in the algorithms we presented, to focus on collisions, indeed reflect experimental results for the establishment of new cell states using epigenetics.

A desired property of cellular and molecular systems is that they be self-stabilizing. That is, they should be able to recover and restore their original state after a disturbance (a transit failure) without any outside intervention. We note that our consensus write-erase algorithm is self-stabilizing. It can easily tolerate a limited number of arbitrary cell changes and processor failures.

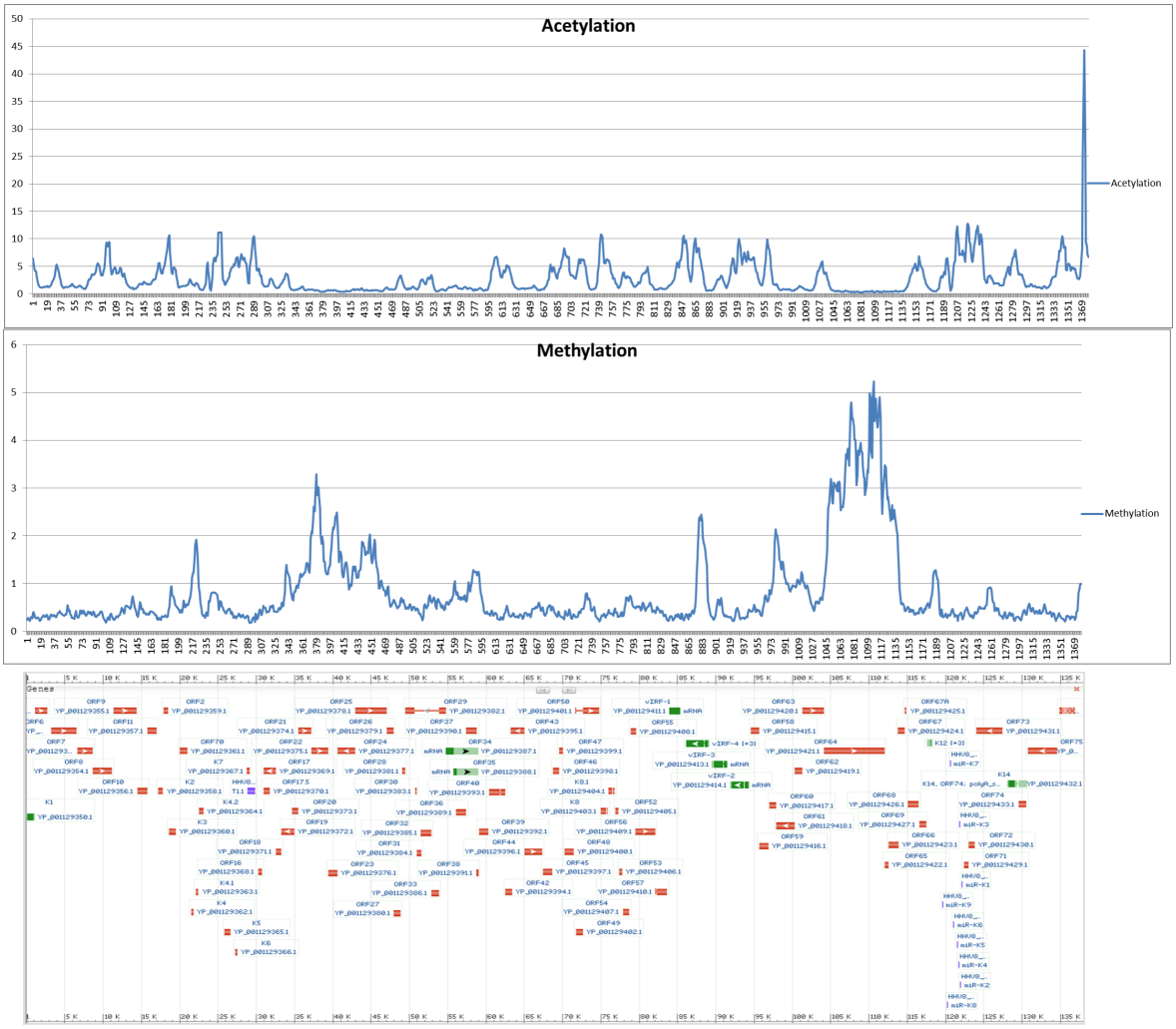

Figure 5.4: Different histone modifiers competing for the same residue, Histone 3 Lysine 9 (H3K9). Latent Kaposi Sarcoma-Associated Herpesvirus Genomes (Resolution 250 bp). Two types of histone modifiers are competing to put acetylation and methylation marks on the residue. We can see stretches of regions with only methylation or acetylation marks, showing regional consensus.

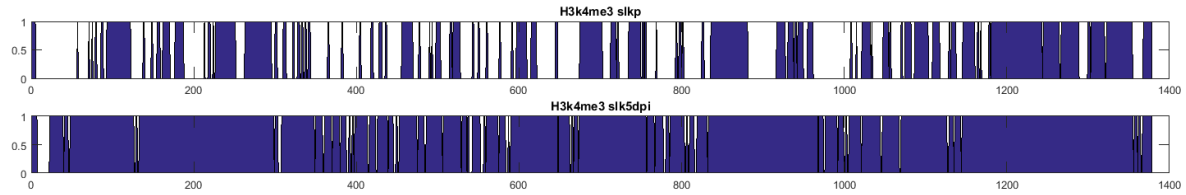

Figure 5.5: Increase in histone modification intensity from the initial stage of infection (slkp) to 5 days post infection (5dpi). We can see that the later time point has higher overall consensus of methylatio histone mark.

Our consensus algorithm is "one-shot," and we would also like to cover the "long-lived" version in which we may switch again possibly many times (i.e., repeated consensus). Also, we assume that all the shared memory locations are initially empty and it would be nice to be able to remove this assumption, especially when a need to establish a new state arises. A tiny change to our algorithm, seems to achieve the above. This is done by assuming that a v-eraser "once in a while" (i.e., at random) unconditionally erase v, even when there is no collision. In such case, if the ratio between the number of 0-erasers and 1-erasers changes significantly, the decision value will change as well. Thus this version can move out of a full consensus state.

There are several, possibly faster, variants of our consensus algorithm that are theoretically interesting, but require making assumptions that are not acceptable from a biological standpoint. For example, we may assume that each writer is in one of two states: active or inactive. Initially, all writers are active. In an active state, a writer behaves as before (scans and writes in empty locations). In an inactive state, a writer scans the array but never writes. Let k_1 and k_2 be small positive integers. An active (resp. inactive) v-writer become inactive (resp. active) if the value in all the last k_1 (resp. k_2) locations it has visited is $1 - v$ (resp. v). Such a solution requires each writer to have a few additional bits of local memory, which is biologically unrealistic.

Chapter 6

Conclusion and Future Work

6.1 Summary of contribution

Computer science and biology have enjoyed a long and useful bi-directional relationship for decades. In this thesis, we discuss many parallel requirements and techniques shared by biological and computational systems. Often times thinking computationally about biological processes leads to more accurate models. Particularly, we show both theoretically and experimentally that decentralized/ distributed biological systems can be formulated by adapted versions of some of the well known distributed computational algorithms. Bacterial chemotaxis and genome wide epigenetic modification are the two biological processes we discuss in this thesis. The first is formulated as a modified version of distributed gradient descent algorithm and the second is formulated as a shared memory consensus algorithm. We demonstrate that several biologically inspired modifications actually improves the performance of these computational algorithms in real life applications with constrained resources and noisy environments. We also designed and conducted several experiments to corroborate the accuracy of the proposed models and show that it significantly improves upon prior models in explaining and/or predicting the biological systems of interest. Following are the key contributions of this thesis.

- **Bacteria swarm food search as distributed gradient descent process:**

- We proposed a novel formulation of cell-cell communication of bacteria swarm. We show that a distributed gradient descent model can efficiently interpret the communication of agents under restricted resources and complex environment (such as low message complexity, agent anonymity, and dynamic topology).
- We have proved that, under reasonable biological assumptions, the DGD model is guaranteed to converge. This explains how bacteria can efficiently coordinate food search in harsh environments.
- Social dynamics of bacteria can also affect their ability to resist antimicrobial therapy, and we have also shown in simulation that the inhibition of bacterial cooperation is an alternative approach to minimize collective resistance.

- **Bacterial DGD as mass evacuation strategy:**
  - We proposed an extension of the bacterial based Distributed Gradient Descent (DGD) model for mass scale evacuations. This is the first evacuation strategy proposed for efficient emergency evacuation in a dynamic scenario with unforeseen obstacles.
  - We show in simulation of the 8th floor plan of Gates that the proposed strategy outperforms the prior models of evacuation when unknown obstacles are added in the topology.
- **Adaptive tumbling in bacterial chemotaxis on obstacle-laden terrains**
  - We obtained new biological insight on how *E. coli* cells' adapt to obstacle-laden terrain. We show that the size or density of evenly spaced obstacles do not alter the average exit rate of *E. coli* cells from microchambers in response to external attractants.
  - We also show, both by analyzing the revised DGD model and by experimentally following single cells, that the reduced exit time in the presence of obstacles is a consequence of reduced tumbling frequency that is adjusted by the *E. coli* cells in

response to the topology of their environment.

  - These findings imply operational short-term memory of bacteria while moving through complex environments in response to chemotactic stimuli and motivate improved algorithms for self-autonomous robotic swarms.

- **Genome Wide epigenetic modifications as a shared memory consensus problem**
  - To the best of our knowledge, this thesis is the first attempt to study biologically inspired distributed computing algorithms in the context of a molecular shared memory systems.
  - We formulated a consensus problem for the histone modifiers regulating chromatin states and presented algorithms and analyzed their expected run time both in theory and simulation.
  - Experimentally validate the predictions of shared memory consensus algorithm.

6.2 Future work

6.2.1 Bacterial chemotaxis and DGD

While our work has already addressed several issues related to the biological processes we studied and their computational counterparts, there are still several extensions we would like to study. We have shown application of bacterial based DGD model in designing mass evacuation strategy. However, the extensions of the model could have potential application in many other computational systems as well, such as sensor networks, mobile /cellular networks etc. While the discussion and simulations in the thesis focused on a synchronous model, the method can be extended to asynchronous execution as well, since many distributed systems actually perform asynchronously (including bacteria cells). The next step in theoretical analysis would be to prove convergence for such asynchronous models. There has been some recent interest in us-

ing variants of DGD to improve the ability of distributed search algorithms aimed at finding various distributions [31]. The advantage of such methods is that they can often be run on a distributed cluster which helps them use much stronger computing power. However, communication requirements can limit their efficiency. Our method, while trying to address a different problem, may prove useful for such distributed optimization as well since it requires minimal, aggregate messages and can work well even if some of the messages are dropped or received asynchronously. More generally, our results provide additional support to the usefulness of studies that attempt to determine what and how biological processes compute, and, in turn use the results to improve computational methods [2].

In addition to the computational applications, several predictions were made from the simulation of our bacterial DGD model, such as effects of silent population, swarm sizes, terrain etc. While we experimentally verified the effect of obstacle-laden terrain, the other aspects are yet to be investigated. In future we can design and execute experiments to test how bacteria cells adapt their chemotaxis dynamics when mutant cells are present in the population. Mutant cells could represent different types of inhibitory effects, such as that of blind and/or silent cells. Blind cells cannot sense the chemoattractant or cell-cell signaling molecules while silent cells cannot secrete the signaling molecules needed for cell-cell communication. The signaling pathway of bacterial chemotaxis has been extensively studied in E. coli cells. E. coli cells can sense chemoattractant gradients through several receptors such as Tsr, Tar, Tap, Trg, and Aer. These are usually clustered at the bacterial poles, Tsr and Tar being the two of the most abundant ones. These chemoreceptors sense extracellular (chemoattractant and/or cell-cell signaling) molecules and utilize several of cytoplasmic signaling proteins to control movement and sensory adaptation[20]. An interesting study would be to test how the population dynamics change when mutant cells with disabled Tsr and/or Tar receptors (blind cells) are present along with the wild type cells. By plotting the escape time as a function of mutant cell fraction we can quantify the effect and verify the corresponding model predictions. In addition to that, we can further inves-

tigate the possible signaling pathway of secretion of signaling molecules by cells. This might enable us to verify our model prediction about silent cells with experiments on cell migration of mutant cells that are unable to secrete the signaling proteins.

6.2.2 Shared memory models

Even though our proposed consensus algorithm is based on biological insight regarding the long stretches of common modifications, we have not experimentally tested any of the predictions of the algorithm. While we showed example of regional consensus on bulk histone modification data, it does not fully capture how such consensus is achieved at individual DNA basepair resolution. To investigate that, we need to perform chip seq experiment at a single cell level. There is on going research on developing techniques to get reliable histone modification tracks from single cells. Hopefully with the advent of the new technology we would be able to validate the proposed model. Computationally, there is room to achieve a tighter bound on the runtime of the proposed consensus algorithm. Possible formulation of number of collisions as a function of time could achieve such tighter bound. While this thesis is the first to formulate a consensus problem for genomewide epigenetic modification and propose possible algorithms, there is still room for possible runtime improvement with modified versions of the proposed algorithm. However, consistency with biological assumptions, proof of correctness, and theoretical runtime bound are the key aspects to consider while proposing new algorithm for shared memory consensus of histone modification in DNA.

6.2.3 Other biologically distributed algorithms

We have discussed several algorithmic principles that are shared by both biological and computational system, such as message passing, feedback, randomness. However, there are many different aspects of distributed computing that warrants new models. Many distributed computing algorithms have dynamic node/edge deletion/addition, which the current biology inspired

models do not address. Similarly there are different biological systems that employ hierarchical distributed computing such as gene regulation and neuronal networks. These properties are actually conserved across different species, suggesting their efficiency and robustness in computation. We can analyze and build new computational models from these systems as well. Evolution has shaped these biological systems for optimal performance under constrained resources, often trading-off between efficiency and robustness. These criteria are essential in man made distributed systems as well. Hence, additional biological-computational bi-directional studies can actually provide basis for new computational algorithms and at the same time improve our understanding of nature.

Bibliography

[1] C. Nak-Young and Y-J Cho. *Distributed Autonomous Robotic Systems: The 12th International Symposium*, volume 112. Springer, 2016. 1.1

[2] S. Navlakha and Z. Bar-Joseph. Distributed information processing in biological and computational systems. *Communications of the ACM*, 58(1):94–102, 2015. 1.1, 5.6, 6.2.1

[3] Jonas Nathan Schwertfeger and Odest Chadwicke Jenkins. Multi-robot belief propagation for distributed robot allocation. In *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, pages 193–198. IEEE, 2007. 1.1

[4] István Hegedűs, Márk Jelasity, Levente Kocsis, and András A Benczúr. Fully distributed robust singular value decomposition. In *Peer-to-Peer Computing (P2P), 14-th IEEE International Conference on*, pages 1–9. IEEE, 2014. 1.1

[5] Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 513–522. ACM, 2010. 1.1

[6] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982. 1.1

[7] C. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995. 1.1

[8] B. Prabhakar, K. N. Dektar, and D. M. Gordon. The regulation of ant colony foraging

activity without spatial information. *PLoS computational biology*, 8(8):e1002670, 2012. 1.1, 1.1.1

[9] L. Buesing, J. Bill, B. Nessler, and W. Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology*, 7(11):e1002211, 2011. 1.1, 1.1.1

[10] A. Teroand, R. Kobayashi, and T. Nakagaki. Physarum solver: A biologically inspired method of road-network navigation. *Physica A: Statistical Mechanics and its Applications*, 363(1):115–119, 2006. 1.1, 1.1.1

[11] A. Destexhe and D. Contreras. Neuronal computations with stochastic network states. *Science*, 314(5796):85–90, 2006. 1.1.1

[12] Sanjoy Dasgupta, Charles F Stevens, and Saket Navlakha. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796, 2017. 1.1.1

[13] S. A. Smolka, A. Tiwari, L. Esterle, A. Lukina, J. Yang, and R. Grosu. Attacking the v: On the resiliency of adaptive-horizon mpc. *arXiv preprint arXiv:1702.00290*, 2017. 1.1.1

[14] Rion G Taylor and Roy D Welch. Chemotaxis as an emergent property of a swarm. *Journal of bacteriology*, 190(20):6811–6816, 2008. 1.2

[15] George H Wadhams and Judith P Armitage. Making sense of it all: bacterial chemotaxis. *Nature Reviews Molecular Cell Biology*, 5(12):1024–1037, 2004. 1.2, 1.2

[16] Remy Colin and Victor Sourjik. Emergent properties of bacterial chemotaxis pathway. *Current opinion in microbiology*, 39:24–33, 2017. 1.2

[17] Jerome Wong-Ng, Antonio Celani, and Massimo Vergassola. Exploring the function of bacterial chemotaxis. *Current opinion in microbiology*, 45:16–21, 2018. 1.2

[18] Rasika M Harshey, Ikuro Kawagishi, Janine Maddock, and Linda J Kenney. Function, diversity, and evolution of signal transduction in prokaryotes. *Developmental cell*, 4(4):459–465, 2003. 1.2, 4.2.2

[19] Christopher M Waters and Bonnie L Bassler. Quorum sensing: cell-to-cell communication in bacteria. *Annu. Rev. Cell Dev. Biol.*, 21:319–346, 2005. 1.2, 2.1, 4.2.2, 4.3.2

[20] Zhicheng Long, Bryan Quaife, Hanna Salman, and Zoltán N Oltvai. Cell-cell communication enhances bacterial chemotaxis toward external attractants. *Scientific reports*, 7(1):12855, 2017. 1.2, 4.2.3, 4.5.3, 4.6, 6.2.1

[21] Marcus J Tindall, Philip K Maini, Steven L Porter, and Judith P Armitage. Overview of mathematical approaches used to model bacterial chemotaxis ii: bacterial populations. *Bulletin of mathematical biology*, 70(6):1570, 2008. 1.2

[22] Sibylle D Muller, Jarno Marchetto, Stefano Airaghi, and P Kournoutsakos. Optimization based on bacterial chemotaxis. *IEEE transactions on Evolutionary Computation*, 6(1):16–29, 2002. 1.2

[23] Hanning Chen, Yunlong Zhu, and Kunyuan Hu. Cooperative bacterial foraging optimization. *Discrete Dynamics in Nature and Society*, 2009, 2009. 1.2

[24] I Richard Lapidus and Ralph Schiller. A mathematical model for bacterial chemotaxis. *Biophysical journal*, 14(11):825–834, 1974. 1.2

[25] Evelyn F Keller and Lee A Segel. Model for chemotaxis. *Journal of theoretical biology*, 30(2):225–234, 1971. 1.2, 4.1, 4.2.1, 4.2, 4.3, 4.4, 4.5.1

[26] Adi Shklarsh, Gil Ariel, Elad Schneidman, and Eshel Ben-Jacob. Smart swarms of bacteria-inspired agents with performance adaptable interactions. *PLoS computational biology*, 7(9):e1002177, 2011. 1.2, 2.1, 2.2.1, 2.2.2, 2.2.3, 2.2.5, 2.3.1, 3.2.1, 3.3.2, 4.1, 4.2.1, 4.2.2, 4.2.3, 4.2, 4.3, 4.4, 4.4.2, 4.5.1, 4.5.3

[27] S. Singh, S. Rashid, Z. Long, S. Navlakha, H. Salman, Z. N. Oltvai, and Z. Bar-Joseph. Distributed gradient descent in bacterial food search. In *Proceedings of the 20th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, 2016. 1.2, 1, 4.1, 4.2.3, 4.2, 4.3, 4.5.1, 4.5.6

[28] Howard C Berg and Douglas A Brown. Chemotaxis in escherichia coli analysed by three-dimensional tracking. *Nature*, 239(5374):500, 1972. 1.2, 4.1, 4.2.1, 4.2, 4.3, 4.4, 4.5.1, 4.5.6

[29] Colin Torney, Zoltan Neufeld, and Iain D Couzin. Context-dependent interaction leads to emergent search behavior in social aggregates. *Proceedings of the National Academy of Sciences*, 106(52):22055–22060, 2009. 1.2, 2.1

[30] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48, 2009. 1.2

[31] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017. 1.2, 6.2.1

[32] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995. 1.2.1, 3.1, 3.3, 3.3.2

[33] Zarita Zainuddin, M Shuaib, and I Abu-Sulyman. The characteristics of the factors that govern the preferred force in the social force model of pedestrian movement. In *Asian Mathematical Conference*, 2009. 1.2.1, 3.1, 3.3, 3.3.2

[34] Lei Hou, Jian-Guo Liu, Xue Pan, and Bing-Hong Wang. A social force evacuation model with the leadership effect. *Physica A: Statistical Mechanics and its Applications*, 400:93–99, 2014. 1.2.1, 3.4

[35] Mingliang Xu, Yunpeng Wu, Pei Lv, Hao Jiang, Mingxuan Luo, and Yangdong Ye. misfm: On combination of mutual information and social force model towards simulating crowd evacuation. *Neurocomputing*, 168:529–537, 2015. 1.2.1

[36] Ning Guo, Rui Jiang, Mao-Bin Hu, Jian-Xun Ding, and Zhong-Jun Ding. Escaping in couples facilitates evacuation: Experimental study and modeling. *arXiv preprint arXiv:1512.05120*, 2015. 1.2.1, 3.1, 3.3

[37] Yanbin Han and Hong Liu. Modified social force model based on information trans-

mission toward crowd evacuation simulation. *Physica A: Statistical Mechanics and its Applications*, 469:499–509, 2017. 1.2.1, 3.1, 3.3

[38] Hans Hardmeier, Andrin Jenal, Beat Küng, and Felix Thaler. Lecture with computer exercises: Modelling and simulating social systems with matlab. 2012. 1.2.1, 3.1, 3.3, 3.3.2

[39] P. M. Diesinger and D. W. Heermann. Depletion effects massively change chromatin properties and influence genome folding. *Biophysical journal*, 97(8):2146–2153, 2009. 1.3

[40] Shelley L Berger. The complex language of chromatin regulation during transcription. *Nature*, 447(7143):407, 2007. 1.3

[41] T. Günther and A. Grundhoff. The epigenetic landscape of latent kaposi sarcoma-associated herpesvirus genomes. *PLoS pathogens*, 6(6):e1000935, 2010. 1.3, 1.3, 5.1, 5.5

[42] C. L. Peterson and M. Laniel. Histones and histone modifications. *Current Biology*, 14(14):R546–R551, 2004. 1.3, 5.1

[43] Nathaniel A Hathaway, Oliver Bell, Courtney Hodges, Erik L Miller, Dana S Neel, and Gerald R Crabtree. Dynamics and memory of heterochromatin in living cells. *Cell*, 149(7):1447–1460, 2012. 1.3

[44] Karl Abrahamson. On achieving consensus using a shared memory. In *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, pages 291–302. ACM, 1988. 1.3.1

[45] J. Aspnes and M. Herlihy. Fast randomized consensus using shared memory. *Journal of algorithms*, 11(3):441–461, 1990. 1.3.1, 5.3

[46] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001. 2.1

[47] Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009. 2.1

[48] John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986. 2.1, 2.2.4, 2.2.4, 2.2.4, 2.2.4, 3

[49] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010. 2.2.1

[50] Fanglin Li, Bin Wu, Liutong Xu, Chuan Shi, and Jing Shi. A fast distributed stochastic gradient descent algorithm for matrix factorization. In *Proceedings of the 3rd International Conference on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications-Volume 36*, pages 77–87. JMLR. org, 2014. 2.2.1

[51] C. Wang J. Chiaochun I. Nemenman R. Cheong, A. Rhee and A. Levchenko. Information transduction capacity of noisy biochemical signaling networks. *Science*, 334(6054):354–358, 2011. 2.2.1, 2.2.3

[52] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014. 2.2.1

[53] Maximilian Weitz, Andrea Muckl, Korbinian Kapsner, Ronja Berg, Andrea Meyer, and Friedrich C Simmel. Communication and computation by bacteria compartmentalized within microemulsion droplets. *Journal of the American Chemical Society*, 136(1):72–75, 2013. 2.2.3

[54] Vic Norris, Abdallah Zemirline, Patrick Amar, Jean Nicolas Audinot, Pascal Ballet, Eshel Ben-Jacob, Gilles Bernot, Guillaume Beslon, Armelle Cabin, Eric Fanchon, et al. Computing with bacterial constituents, cells and populations: from bioputing to bactoputing.

Theory in Biosciences, 130(3):211–228, 2011. 2.2.3

[55] Ortrud Wartlick, Anna Kicheva, and Marcos González-Gaitán. Morphogen gradient formation. *Cold Spring Harbor perspectives in biology*, 1(3):a001255, 2009. 2.2.3

[56] Yuval Emek and Roger Wattenhofer. Stone age distributed computing. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 137–146. ACM, 2013. 2.2.3, 2.3.1, 3.2.3, 3.3.4, 4.2.2

[57] Burkhard A Hense, Christina Kuttler, Johannes Müller, Michael Rothballer, Anton Hartmann, and Jan-Ulrich Kreft. Does efficiency sensing unify diffusion and quorum sensing? *Nature Reviews Microbiology*, 5(3):230, 2007. 2.2.4

[58] Nikita Vladimirov, Linda Løvdok, Dirk Lebiedz, and Victor Sourjik. Dependence of bacterial chemotaxis on gradient shape and adaptation rate. *PLoS computational biology*, 4(12):e1000242, 2008. 2.2.4

[59] Nancy A Lynch. *Distributed algorithms*. Elsevier, 1996. 2.2.4

[60] Karl Francis and Bernhard O Palsson. Effective intercellular communication distances are determined by the relative time constants for cyto/chemokine secretion and diffusion. *Proceedings of the National Academy of Sciences*, 94(23):12258–12262, 1997. 2.2.4

[61] Eugene A Yurtsev, Hui Xiao Chao, Manoshi S Datta, Tatiana Artemova, and Jeff Gore. Bacterial cheating drives the population dynamics of cooperative antibiotic resistance plasmids. *Molecular systems biology*, 9(1):683, 2013. 2.3, 2.3.3

[62] Gregory J Velicer, Lee Kroos, and Richard E Lenski. Developmental cheating in the social bacterium myxococcus xanthus. *Nature*, 404(6778):598, 2000. 2.3.3

[63] Katrin Hammerschmidt, Caroline J Rose, Benjamin Kerr, and Paul B Rainey. Life cycles, fitness decoupling and the evolution of multicellularity. *Nature*, 515(7525):75, 2014. 2.3.3

[64] Joan E Strassmann, Yong Zhu, and David C Queller. Altruism and social cheating in the social amoeba dictyostelium discoideum. *Nature*, 408(6815):965, 2000. 2.3.3

[65] Nicole M Vega and Jeff Gore. Collective antibiotic resistance: mechanisms and implications. *Current opinion in microbiology*, 21:28–34, 2014. 2.4

[66] Massimo Vergassola, Emmanuel Villermaux, and Boris I Shraiman. infotaxis as a strategy for searching without gradients. *Nature*, 445(7126):406, 2007. 2.4

[67] Sabrina Rashid, Shashank Singh, Saket Navlakha, and Ziv Bar-Joseph. A bacterial based distributed gradient descent model for mass scale evacuations. *Swarm and Evolutionary Computation*, 46:97–103, 2019. 1

[68] Jian Ma, WG Song, W Tian, Siu Ming Lo, and GX Liao. Experimental study on an ultra high-rise building evacuation in china. *Safety science*, 50(8):1665–1674, 2012. 3.1

[69] Zhiming Fang, Weiguo Song, Jun Zhang, and Hao Wu. Experiment and modeling of exit-selecting behaviors during a building evacuation. *Physica A: Statistical Mechanics and its Applications*, 389(4):815–824, 2010. 3.1

[70] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2013. 3.2.3

[71] Predrag Tosic and Gul Agha. Maximal clique based distributed group formation for autonomous agent coalitions. *A PARAMETRIC MODEL FOR LARGE SCALE AGENT SYSTEMS*, page 195, 2005. 3.2.3

[72] Sabrina Rashid, Zhicheng Long, Shashank Singh, Maryam Kohram, Harsh Vashistha, Saket Navlakha, Hanna Salman, Zoltan N. Oltvai, and Ziv Bar-Joseph. Adjustment in tumbling rates improves bacterial chemotaxis on obstacle-laden terrains. *Proceedings of the national academy of sciences, in press*, 2019. 1

[73] Amir M Naghsh, Jeremi Gancet, Andry Tanoto, and Chris Roast. Analysis and design of human-robot swarm interaction in firefighting. In *RO-MAN 2008-The 17th IEEE International Symposium on Robot and Human Interactive Communication*, pages 255–260.

IEEE, 2008. 4.1, 4.5.6, 4.6

[74] Gábor Vásárhelyi, Csaba Virágh, Gergő Somorjai, Tamás Nepusz, Agoston E Eiben, and Tamás Vicsek. Optimized flocking of autonomous drones in confined environments. *Science Robotics*, 3(20):eaat3536, 2018. 4.1

[75] RN Bearon and TJ Pedley. Modelling run-and-tumble chemotaxis in a shear flow. *Bulletin of mathematical biology*, 62(4):775–791, 2000. 4.2.1, 4.2, 4.3, 4.4, 4.5.1

[76] Anne E Carpenter, Thouis R Jones, Michael R Lamprecht, Colin Clarke, In Han Kang, Ola Friman, David A Guertin, Joo Han Chang, Robert A Lindquist, Jason Moffat, et al. Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology*, 7(10):R100, 2006. 4.3.3

[77] Khuloud Jaqaman, Dinah Loerke, Marcel Mettlen, Hirotaka Kuwata, Sergio Grinstein, Sandra L Schmid, and Gaudenz Danuser. Robust single-particle tracking in live-cell time-lapse sequences. *Nature methods*, 5(8):695, 2008. 4.3.3

[78] Xiao-Lun Wu and Albert Libchaber. Particle diffusion in a quasi-two-dimensional bacterial bath. *Physical review letters*, 84(13):3017, 2000. 4.4.2

[79] Saket Navlakha and Ziv Bar-Joseph. Algorithms in nature: the convergence of systems biology and computational thinking. *Molecular systems biology*, 7(1):546, 2011. 4.5.2, 4.6

[80] Atsushi Tero, Seiji Takagi, Tetsu Saigusa, Kentaro Ito, Dan P Bebber, Mark D Fricker, Kenji Yumiki, Ryo Kobayashi, and Toshiyuki Nakagaki. Rules for biologically inspired adaptive network design. *Science*, 327(5964):439–442, 2010. 4.5.2, 4.6

[81] Peter Galajda, Juan Keymer, Paul Chaikin, and Robert Austin. A wall of funnels concentrates swimming bacteria. *Journal of bacteriology*, 189(23):8704–8707, 2007. 4.5.4

[82] Seth Tisue and Uri Wilensky. Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Boston, MA,

2004. 4.5.6, 4.6

[83] Micael S Couceiro, Patricia A Vargas, Rui P Rocha, and Nuno MF Ferreira. Benchmark of swarm robotics distributed techniques in a search task. *Robotics and Autonomous Systems*, 62(2):200–213, 2014. 4.5.6, 4.6

[84] Micael S Couceiro, Fernando ML Martins, Rui P Rocha, and Nuno MF Ferreira. Mechanism and convergence analysis of a multi-robot swarm approach based on natural selection. *Journal of Intelligent & Robotic Systems*, 76(2):353–381, 2014. 4.5.6, 4.6

[85] María Vela-Pérez, Marco A Fontelos, and S Garnier. From individual to collective dynamics in argentine ants (linepithema humile). *Mathematical biosciences*, 262:56–64, 2015. 4.6

[86] Anke Ordemann, Gabor Balazsi, and Frank Moss. Pattern formation and stochastic motion of the zooplankton daphnia in a light field. *Physica A: statistical mechanics and its applications*, 325(1-2):260–266, 2003. 4.6

[87] Vasily Yu Zaburdaev. Random walk model with waiting times depending on the preceding jump length. *Journal of Statistical Physics*, 123(4):871–881, 2006. 4.6

[88] Juan E Keymer, Peter Galajda, Cecilia Muldoon, Sungsu Park, and Robert H Austin. Bacterial metapopulations in nanofabricated landscapes. *Proceedings of the National Academy of Sciences*, 103(46):17290–17295, 2006. 4.6

[89] Ekaterina Korobkova, Thierry Emonet, Jose MG Vilar, Thomas S Shimizu, and Philippe Cluzel. From molecular noise to behavioural variability in a single bacterium. *Nature*, 428(6982):574, 2004. 4.6

[90] Howard C Berg. *E. coli in Motion*. Springer Science & Business Media, 2008. 4.6

[91] Pushkar P Lele, Basarab G Hosu, and Howard C Berg. Dynamics of mechanosensing in the bacterial flagellar motor. *Proceedings of the National Academy of Sciences*, 110(29):11839–11844, 2013. 4.6

[92] Mahmut Demir and Hanna Salman. Resonance in the response of the bacterial flagellar motor to thermal oscillations. *Physical Review E*, 95(2):022419, 2017. 4.6

[93] Z. Tang, O. J. Luo, X. Li, M. Zheng, J. J. Zhu, P. Szalaj, P. Trzaskoma, A. Magalska, J. Wlodarczyk, B. Ruszczycki, et al. Ctcf-mediated human 3d genome architecture reveals chromatin topology for transcription. *Cell*, 163(7):1611–1627, 2015. 5.2

[94] A. D. Goldberg, D. C. Allis, and E. Bernstein. Epigenetics: a landscape takes shape. *Cell*, 128(4):635–638, 2007. 5.2

[95] R. Cole and O. Zajicek. The APRAM: incorporating asynchrony into the PRAM model. In *SPAA*, pages 169–178, 1989. 5.2

[96] M. Herlihy and N. Shavit. *The Art of Multiprocessor Programming*. Morgan Kaufmann Publishers, 2008. 508 pages. 5.2

[97] M. Raynal. *Concurrent Programming: Algorithms, Principles, and Foundations*. Springer, 2013. 515 pages. 5.2

[98] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980. 5.3

[99] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. 5.3

[100] K. Abrahamson. On achieving consensus using a shared memory. In *Proc. 7th ACM Symp. on Principles of Distributed Computing*, PODC '88, pages 291–302, 1988. 5.3

[101] J. Aspnes, G.Shahand, and J. Shah. Wait-free consensus with infinite arrivals. In *Proc. of the 24th ACM Symposium on Theory of Computing*, STOC '02, pages 524–533, 2002. 5.3

[102] M. J. Fischer, S. Moran, and G. Taubenfeld. Space-efficient asynchronous consensus without shared memory initialization. *Information Processing Letters*, 45(2):101–105, 1993. 5.3

[103] M. C. Loui and H. Abu-Amara. Memory requirements for agreement among unreliable

asynchronous processes. *Advances in Computing Research*, 4:163–183, 1987. 5.3

[104] S. A. Plotkin. Sticky bits and universality of consensus. In *Proc. 8th ACM Symp. on Principles of Distributed Computing*, pages 159–175, 1989. 5.3

[105] M. Saks, N. Shavit, and H. Woll. Optimal time randomized consensus – making resilient algorithms fast in practice. In *Proc. 2nd ACM-SIAM Symposium on Discrete Algorithms*, SODA '91, pages 351–362, 1991. 5.3

[106] D. Dolev, C. Dwork, and L. Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the ACM*, 34(1):77–97, 1987. 5.3

[107] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, 1988. 5.3

[108] M. J. Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *Proceedings of the 1983 International FCT-Conference on Fundamentals of Computation Theory*, 1983. *LNCS 158* Springer Verlag 1983, 127–140. 5.3

[109] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986. 5.3

[110] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985. 5.3

[111] J. Aspnes. Randomized protocols for asynchronous consensus. *Distributed Computing*, 16(2–3):165–175, September 2003. ArXiv version: arXiv:cs.DS/0209014, last updated May 28, 2018. 5.3

[112] I. O. Torres and D. G. Fujimori. Functional coupling between writers, erasers and readers of histone and dna methylation. *Current opinion in structural biology*, 35:68–75, 2015. 5.4.6

[113] S. Panneerdoss, V. K. Eedunuri, P. Yadav, S. Timilsina, S. Rajamanickam, S. Viswanadhapalli, N. Abdelfattah, B. C. Onyeagucha, X. Cui, Z. Lai, et al. Cross-talk among writers,

readers, and erasers of m6a regulates cancer growth and progression. *Science advances*, 4(10):eaar8263, 2018. 5.5

[114] P. Cheung, F. D. Vallania, H. C. Warsinske, M. Donato, S. Schaffert, S. E. Chang, M. Dvorak, C. L. Dekker, M. M. Davis, P. J. Utz, et al. Single-cell chromatin modification profiling reveals increased epigenetic variations with aging. *Cell*, 2018. 5.5