UNIVERSITY OF CALIFORNIA, MERCED

Efficiency in Competitive and Cooperative Foraging

A Dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in

Physics

by

Farnaz Golnaraghi

Committee in charge:

Professor Linda Hirst, Chair
Professor Bin Liu
Professor Ajay Gopinathan

December 2021

The dissertation of Farnaz Golnaraghi, titled Efficiency in Competitive and Cooperative Foraging, is approved, and it is acceptable in quality and form for publication.

(Professor Ajay Gopinathan) Principal Advisor	Date
(Professor Linda Hirst) Committee Chair	Date
(Professor Bin Liu) Committee Member	Date

University of California, Merced
2022

To my parents, Farzaneh Fazeli and Abbas Golnaraghi

Acknowledgments

I am very privileged to have many amazing people in my life. Without them, I wouldn't have been able to write this dissertation! I want to thank my advisor, Ajay Gopinathan. Working with Ajay was a great experience and better than anything I could ever imagine. He is an outstanding physicist who challenges his students intellectually while supporting them through difficulties. I would like to thank my committee members, Prof. Linda Hirst and Prof. Bin Liu for their guidance and valuable feedback on my dissertation. I would like to thank my former and current lab-mates, Katie Copenhagen, Jose Zamora, Ritwika VPS, and Niranjan Sarpangala for all the engaging discussions we had about physics. I would like to acknowledge my collaborators, Patrick Noerr and Kinjal Dasbiswas. I am thankful to David Quint for being a great mentor, and for all the teaching, support and intense discussions that really guided me through this journey.

I also want to thank my friends at UC Merced who helped me throughout the past five years, Alison Huff, Bryan Maelfeyt, Jessica Wilson, Ahmad Alhares, Suhani Nagpal and Palak Dugar. I am very grateful for having friends who are in different parts of this globe, but have always supported me: Mina Rashetnia, Farnoosh Attarzadeh, Aynaz Ameri, Pegah Mavaee, and Kiana Mostaghasi. I also want to thank Nazanin Baghaipour. She has been a very close friend of mine since the day I started learning how to read and write. I feel most indebted to my mom and dad, Farzaneh and Abbas. I never felt emotionally distant from them. Even though we are 7000 miles apart, and I haven't seen them for more than four years! They made many sacrifices to help me grow and follow my goals. Finally, I want to thank my husband, Imtiaz, for all the moral support, and being the best lab-mate I could ever ask for! This research has been funded by the following sources:

- NSF-CREST: Center for Cellular and Biomolecular Machines at UC Merced

(NSF-HRD-1547848)

Farnaz Golnaraghi

136 N. Spruce Ave, South San Francisco, CA 94080
Call: (650)-796-1198 , Email: fgolnaraghi@ucmerced.edu

EDUCATION

PhD Candidate in Physics
University of California, Merced, August 2016 - Present
Current research: Efficiency in Competitive and Cooperative Group Foraging
Supervisor: Prof. Ajay Gopinathan

Master of Science in Physics
University of California, Merced, December 2019

Bachelor of Science in Physics
University of Tehran, Tehran, Iran, June 2016

SELECTED COURSEWORK

Numerical Methods, Partial Diff. Eqs., Scientific Computing, Statistical Mechanics, Measurements & Uncertainties, Biophysics, Intelligent Adaptive Systems, C++ Programming, Molecular Dynamics, Machine Learning

COMPUTER SKILLS

Language & Software: C/C++, Python, Bash, MATLAB, R, LaTeX
Operating Systems: Unix, MS-Windows.

SELECTED PROJECTS

- Optimal search strategies for competing agents (object-oriented with Python)
  - Modeling the foraging patterns of territorial competitors such as tigers and birds using statistical mechanics principles.
  - Developed large scale agent-based Monte Carlo simulation code in Python.
  - Utilized Python's object-oriented features and scientific libraries such as NumPy and SciPy.
  - Utilized ray tracing techniques to optimize the search algorithm and increase the robustness of the program.
- Modeling the collective foraging of malignant lymphocytes (object-oriented with C++)
  - Modeling the behavior of cells as a N-body mass-spring system.
  - Analyzed experimental data using MATLAB.
  - Developed agents based Monte Carlo simulation code in C++.
  - Utilized and debugged pre-written relevant code by former students
  - Developed Bash and Python pipelines to run several simulations, and analyze the results using NumPy, Pandas and Matplotlib.
- Modeling the emergent multi-cellular network structures (object-oriented with C++)
  - Developed an agent-based code for collaborators
  - Helped the collaborators with coding aspects of the project and data analysis

ACADEMIC SERVICE & PROFESSIONAL EXPERIENCE

- NSF-CREST CCBM Computational Graduate Mentor and System Administrator, Spring 2020 - Fall 2021
- GRAD-EXCEL Peer Mentorship Program, Fall 2019 - Spring 2021
- NSF-CREST CCBM C-SIP Mentorship Program, Summer 2019
- Teaching Assistant, Introductory Physics I & II, Calculus I & II, Fall 2016 - Spring 2021

HONORS & AWARDS

- UC Merced Physics Department Summer Fellowship (Summer 2018, 2019, 2020,2021)
- APS, Division of Biological Physics Student Travel Grant (Spring 2020)
- NSF-CREST CCBM Scholar (Fall 2018 - Present)
- NSF-NRT IAS Fellow (AY 2017-2018)
- NSF-NRT ICGE Fellow (Spring 2017)
- Member of National Organization for Development of Exceptional Talents (NODET)

PUBLICATIONS

- "Efficient foraging strategies for territorial competitors", Farnaz Golnaraghi, David A. Quint, Ajay Gopinathan, submitted to Journal of Theoretical Biology, 2021
- "Collective foraging of cells", Farnaz Golnaraghi, Ajay Gopinathan, in prep, 2021
- "Optimizing network formation on elastic substrates", Patrick S. Noerr, Farnaz Golnaraghi, Kinjal Dasbiswas, Ajay Gopinathan, in prep, 2021

CONFERENCE PRESENTATIONS

- Efficiency in competitive group foraging, APS March meeting, Spring 2021
- Clustering dynamics of collectively migrating malignant lymphocytes, CEMB Mechanobiology Symposium, 2021
- Optimal foraging strategies for territorial competitors, APS Far West annual meeting, Fall 2018
- Optimizing the success of multi-agent search, APS March meeting, Spring 2018
- Optimizing success of random searches in various search environments, APS Far West annual meeting, Fall 2017

Abstract

Bacteria, eukaryotic cells, multicellular organisms, animals and even humans forage to find resources such as food. Foraging involves random search processes, and it has been shown that trajectories of many types of foragers resemble random walks. It is also known that solitary foragers can change their foraging strategy to maximize their encounter rate or search efficiency. While there has been much work done on foraging individually, less is known about foraging in groups. Members of a group can cooperate to find resources like herds of Mongolian gazelles, or compete with each other like tigers. The effect of cooperation and competition on foraging patterns is not yet fully understood. In this dissertation, we aim to increase our understanding by focusing on two specific cases - (i) efficient foraging strategies for territorial competitors and (ii) cooperative foraging in cell clusters.

Many animals such as albatrosses are known to exhibit foraging patterns where the distances they travel in a given direction are drawn from a heavy-tailed Lévy distribution. Previous studies have shown that, under sparse resource conditions, solitary foragers perform a maximally efficient search with Lévy exponent equal to 2. However, in nature, there also exist situations where multiple foragers interact with each other competitively. To understand the effects of competition, we develop a stochastic agent-based simulation that models competitive foraging among territorial individuals by incorporating a territory of a certain size around each forager which is not accessible by other competitors. Our results show that with increasing size of the territory and number of agents the optimal Lévy exponent is still approximately 2 but the efficiency of the search decreases except at low values of the Lévy exponent, where increased territories unavailable for searching increases efficiency. Moreover, we show that the variance among the efficiencies of the agents increases with increasing Lévy exponent. Thus, by performing more localized searches, foragers might increase

the mean efficiency of a population, but at the risk of increasing variance in efficiency among individuals. On the other hand, performing more de-localized, smaller Lévy exponent searches can decrease variance but the decrease in efficiency may be countered by increased territorial competition.

While competition is common, many living organisms such as bees, school of fish and caterpillars collaborate with each other to complete a task. Such collaboration extends to the cellular scale. Multicellular aggregates such as cell clusters and tissues exhibit collective migration with complex emergent behaviors that are critical for function and very different from the behavior of the constituent single cells. We focus on the chemotaxis of clusters of malignant lymphocytes, responsible for the metastases of lymphomas. Previous work has shown that, in high chemical gradients, clusters travel towards higher concentrations of chemoattractants while individual cells are repelled and travel in the opposite direction. It was also shown that these clusters show a number of novel collective phases including running, rotating and random phases that were speculated to avoid chemo-repulsion and enhance their chemotactic efficiency. Using agent-based simulations we showed that the chemotactic efficiency of clusters increases with increasing chemical gradient and persists even at high gradients while individual cells and small clusters experience chemo-repulsion at high gradients, in agreement with experiments. Moreover, the internal dynamics of the clusters in the running phase showed three well-defined structures including single vortices, double vortices and disordered structures which we were able to identify in experiments as well. These dynamical states let cells switch between the clusters' rim and core avoiding chemo-repulsion, and allowing the cluster to forage for chemoattractants more consistently.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Overview

In nature, foraging processes are ubiquitous. Living organisms forage to find resources such as food to survive or to reproduce by mating (Fig 1.1). Therefore, the underlying motivation for their movement is to increase their encounters with such resources. Even though, their movement might be affected by many factors such as changes in temperature, density of other organisms, climate and escaping from predators.

Encounters of foragers with resources involve random search processes, and it has been shown that trajectories of foragers can be described by random walks [1]. It was also shown that foragers can tune their random walk statistics to maximize their encounter rate i.e. their search efficiency [2]. The search can be guided by external cues such as cognitive and detection, or it might not be oriented in which it becomes a stochastic process. When location of resources are not known a priori, organisms perform a completely stochastic search [1]. While, when there is knowledge about the location of resources through sensing, memory, or signals from other organisms (e.g. pheromones, bee dances) the search is not considered fully stochastic anymore [5].

Solitary foraging processes, in both cases with and without external cues have been studied well in the literature. On one hand, with no information available to the searcher, in the case of sparse targets, many animals such as albatrosses, jackals [6], and spider monkeys [7] exhibit foraging patterns where distances traveled are drawn from a heavy tailed Lévy distribution [2, 8]. On the other hand, many organisms locate resources using sensory signals. In the presence of external cues and prior

knowledge, the foraging strategy can be modified to make use of the information. Chemotaxis [9], infotaxis [10], and anemotaxis [5] are examples of search processes with external cues. For instance, bacteria, such as E.Coli, use chemotaxis to move towards the source of energy by climbing in the direction of positive gradient [9].

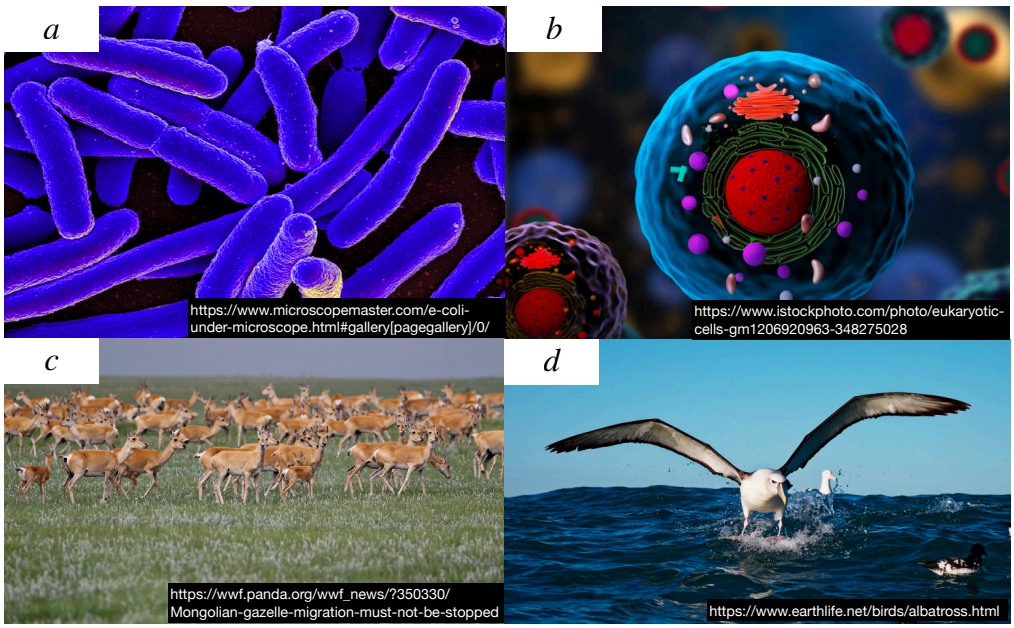

Figure 1.1: (a) E. Coli, (b) Eukaryotic cells, (c) Mongolian gazelles, and (d) Albatrosses forage to find resources such as food or to reproduce by mating [1].

Foraging has been studied by using concepts and techniques in statistical physics. Statistical physics focuses on macroscopic phenomena that result from microscopic interactions between members of a system. Similarly, the trajectories of foragers searching for food in macroscopic scale depends on the steps taken by the organisms with their limited information on microscopic scale [11]. Moreover, foraging involves disorder, randomness, and entropy since foragers are not necessarily aware of the resources locations [12]. Finally, the concepts of universality and scaling in statistical physics can be applied to foraging problems, to see whether diverse organisms in a variety of environments move in similar ways [1].

Many situations involve foraging in groups with cooperative (herds, or clusters of cells) or competitive (territorial animals, or bacteria) interactions. Autonomous organisms, which communicate locally, form groups that are able to behave collectively

such as flocks, swarms, and crowds. These dynamics allow them to complete complicated tasks like foraging, hunting, thermoregulation, and protection against predators [3]. Field studies, theoretical, and computational models have investigated different aspects of foraging in groups and collective behavior. For example, collective foraging is favored in the cases of clumpy and scarce resources, but not when resources are abundant [13]. Foraging time of a group of robots that avoid interfering with each other depends on the group size, and an optimal size is observed where foraging time is minimized [14].

Forming a group can be advantageous for multiple reasons including sharing information and finding resources in a more efficient way [15]. Benefits of searching cooperatively appears in a wide range of settings from robots [16] to living organisms. The group forming also allows individuals to overcome complicated tasks and situations[17]. These behaviors extend to cellular scales, and cells form clusters and migrate collectively to participate in tissue development, tumor metastasis, and repair. They cooperate with each other to complete various tasks in the body.

In this thesis, I am going to address some aspects of collective foraging that have not been studied before:

- What are the effects of territorial competition on foraging efficiency and strategy?
- How clusters of malignant lymphocytes are able to perform chemo-attraction in presence of chemo-repulsion for individual cells?

(i) Group foraging with territorial competitors:

In Chapter §3, I investigate how search strategy changes in the presence of territorial competitors. I explore the different types of strategies that can optimize group foraging in these cases. My focus relies on implementation of different landscapes such that I observe dynamic properties for territorial competitors. I use an agent-based model which utilizes Lévy distributions [18, 19] to address efficient foraging strategies for terrestrial and aerial animals in presence of competition. I study the effects of increasing the population of foragers and their size of territory on search efficiency, and strategies that maximize the encounter rate with resources for destructive and non-destructive foraging. Furthermore, I look at the variance of the efficiency of the foragers, and how it changes based on the search strategy, and number of searchers.

(ii) Foraging in cellular clusters:

In Chapter §4, by using agent-based simulations, I study the intracellular dynamics which lead to robust chemotaxis in clusters of malignant lymphocytes. The model is similar to the one in [20]. However, I add chemo-repulsion to the model which allows the model to not only capture the necessary behavior for robust chemotaxis such as rotating, running and random phases, but also allows the cluster to do chemo-repulsion in very high chemical gradients. I show how the chemotactic efficiency changes as a function of chemical gradient and verify my results by analysis of experimental data. Moreover, I look at the internal dynamics of the clusters to investigate the mechanism behind chemotaxis in clusters.

Chapter 2

Background concepts

2.1 Concepts for chapter §3

2.1.1 Gaussian distribution, Central limit theorem and Lévy distribution

In chapter §3 we look at random walkers with step lengths coming from a Lévy distribution. The Gaussian distribution describes a variety of phenomena at the macroscopic level. The Gaussian or Normal distribution appears in nature ubiquitously due to the wide applicability of the central limit theorem. Central limit theorem indicates that the sum of a large number of independent and identically distributed random variables that have a finite variance converges to a Gaussian distribution. Moreover, the theorem guarantees that the variance of the Gaussian will grow asymptotically linearly in number of random variables. However, it was shown that the Gaussian distribution is a special case of the skew Lévy α stable distributions [1].

$$\phi(t) = \exp[it\nu - |ct|^{\alpha}(1 - i\beta sign(t)\Phi)], \tag{2.1}$$

$$\Phi(t) = \begin{cases} \tan[\alpha\pi/2], if \alpha \neq 1 \\ -(2/\pi)\ln(|t|), if \alpha = 1 \end{cases} \tag{2.2}$$

$$P(S) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp[-itS]\phi(t)dt \tag{2.3}$$

β term is the asymmetry , ν represents a shift and c is a scale. The Lévy index $\alpha \in (0, 2]$ and it is the key parameter in studying Lévy flights which will appear frequently in chapter §3. When $\alpha \in (0, 2)$ the probability density has an asymptotic power law tail with exponent $\mu = \alpha + 1$. When $\alpha = 2$ the distribution will be Gaussian, and when $\alpha = 1$, it will result in Cauchy distribution [1].

The Brownian motion or the normal diffusion is described by a Gaussian distribution, and the mean squared displacement is related to the time linearly:

$$< r^2 >= 2dDt \tag{2.4}$$

Where d is the dimension, D is the diffusion constant and t is time. If the diffusion cannot be described by the normal diffusion, it will be called anomalous diffusion [21]. In a more general form, Hurst exponent (H) can be used to describe the type of diffusion [22, 23] based on the growth of MSD with time.

$$< x^2 >= t^{2H} \tag{2.5}$$

When $H = \frac{1}{2}$ the equation shows normal diffusion, $H > \frac{1}{2}$ corresponds to superdiffusion where the MSD grows super-linearly with time [24], and $H < \frac{1}{2}$ relates to subdiffusion where the MSD grows sub-linearly with time [24]. Lévy flights have superdiffusive behavior, and continuous random walks with pausing times fit in subdiffusive category. The focus of chapter §3 is on superdiffusion and specifically Lévy flights.

When the step size distribution has a power-law tail $p(l) \sim l^{-\mu}$, Lévy flights arise (Fig. 2.1 a). The probability density function for the walker's position converges to a Lévy stable distribution (Eq. (2.1)) with $\alpha = \mu - 1$ with $0 < \alpha \leq 2$, where $\alpha = 2$ is the normal diffusion (Fig. 2.1 b). The Lévy flights have a constant time independent of the flight length. The time can be considered very small, and the velocity is considered infinite [25].The Lévy walks, however, have constant and final velocity [25]. Both Lévy flights and walks have applications in a variety of systems such as foraging and ecology [1], protein folding dynamics [26], climate and atmospheric physics [27, 28], finance [29, 30, 31], optics [32], lasers, [33], and turbulence [34].

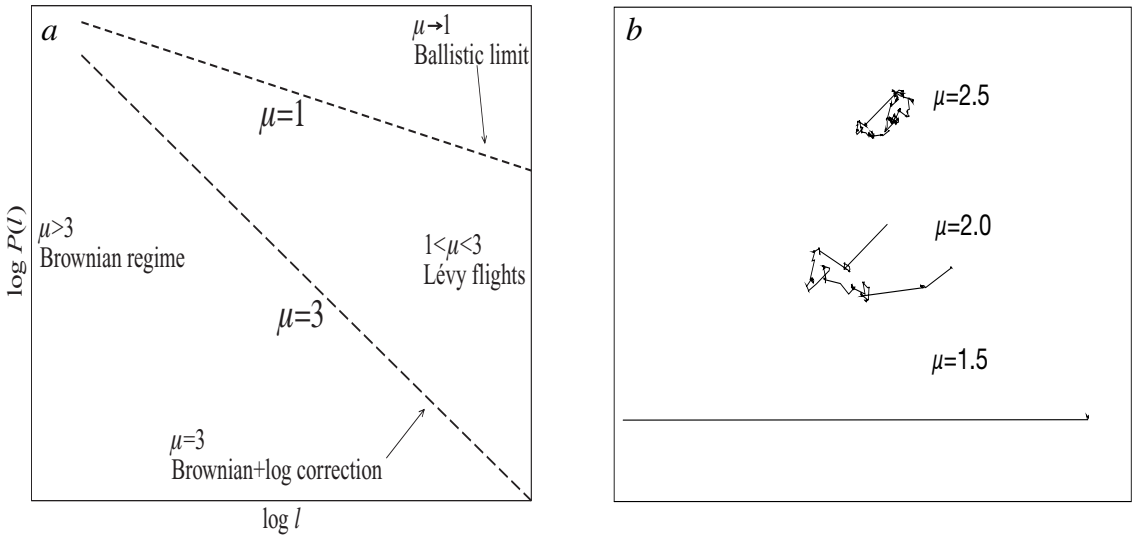

Figure 2.1: (a) Probability density function $P(\ell) \sim \ell^{-\mu}$ for a random walker with step size ℓ. The type of diffusion depends on μ. $\mu \to 1$ corresponds to the ballistic limit and $\mu > 3$ converges to normal diffusion. $1 < \mu \leq 3$ corresponds to Lévy flights and walks. Figure is from [1] (b) Sample trajectories of Lévy flights for different μ values. All of the trajectories have identical total length of 10^3 units. Figure is from [2].

2.1.2 Lévy flights, and optimal foraging strategy for single searchers

It was shown that when the resources are sparse and regenerative, the optimal foraging strategy for solitary foragers is Lévy flights[2, 8]. The efficiency of the search, defined as total target found over total distance traveled, is maximized when the Lévy index, μ is roughly equal to 2. In this section, I cover how it was shown analytically that $\mu = 2$ maximizes the efficiency (Fig. 2.2 c). In chapter §3 this will be validated through simulations.

If the target density is ρ, and forager has a limited perceptive range or vision radius r_v to detect the targets (Fig. 2.2 a,b), the mean free path between targets can be written as $\lambda = (2r_v\rho)^{-1}$ which also appears in (Eq. (3.2)). This relation can be obtained by considering the forager and targets as particles that collide in a rectangle with length of l which is the flight length, and $2 \times r_v$ as the width because the forager detect the targets within radius of r_v, similar to collision of gas particles in a cylinder.

The velocity of the forager is $v = \frac{l}{t}$, and the distance can be written as $l = vt$. Then, the collision rate which is the number of particles over distance traveled will be $\frac{2r_v\rho vt}{vt}$ which simplifies to $2r_v\rho$. The inverse of the collision rate will be the mean free path or λ.

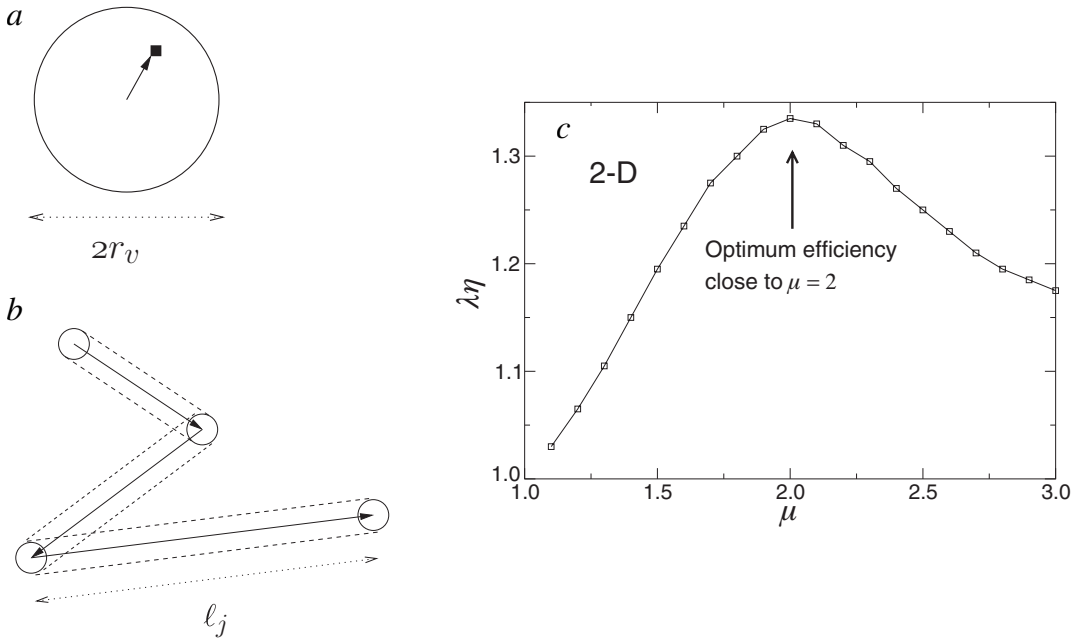

Figure 2.2: Foraging strategy for a Lévy searcher (a) if a target is located within distance r_v of the forager, the foragers moves directly towards the target. (b) If there is no target within r_v the forager picks a random step length l from the Lévy distribution and a random angle. It constantly looks for targets within r_v while traveling along l. If it finds a target, it goes to the target. If not, it repeats (a) and (b). (c) The efficiency of the search as a function of μ for a solitary forager in two dimensions. $\lambda = 5000$ and the efficiency η is multiplied by λ on the y axes. The efficiency has a maximum when $\mu = 2$. Figures are from [1].

Considering the steps coming from a $p(l) \sim l^{-\mu}$ distribution, we can find the average flight length for a single searcher as:

$$< l > \approx \frac{\int_{r_v}^{\lambda} l^{1-\mu}\,\mathrm{d}l + \lambda\int_{r_v}^{\infty} l^{-\mu}\,\mathrm{d}l}{\int_{r_v}^{\infty} l^{-\mu}\,\mathrm{d}l} = (\frac{\mu-1}{2-\mu})(\frac{\lambda^{2-\mu} - r_v^{2-\mu}}{r_v^{1-\mu}}) + \frac{\lambda^{2-\mu}}{r_v^{1-\mu}} \qquad (2.6)$$

In the second term of (Eq. (2.6)), steps greater than the mean free path are approximated to be in average equal to the mean free path [2]. Efficiency, $\eta = \frac{N_{total}}{L_{total}} =$

$\frac{1}{<l>N}$ (Eq. (3.3)), is defined as total target found over total distance traveled or inverse of average flight length, $< l >$ (Eq. (2.6)), times the average number of flights between two successive targets N. The average number of flights for non-regenerative targets/destructive foraging is approximately:[2]:

$$N_d \approx (\frac{\lambda}{r_v})^{\mu-1} \tag{2.7}$$

In the case of non-destructive foraging, or having regenerative targets (Eq. (**??**)) overestimates the number of steps, and $N_n = N_d^{1/2}$. Note that if $\mu = 3$, where normal diffusion or Brownian motion occurs, $N_d \approx (\lambda)^2$ since $r_v \ll \lambda$. Considering the number of steps between targets, N_d, to be similar to time, t, in the normal diffusion MSD relation (Eq. (2.4)), then $N_d = \frac{\lambda^2}{2D}$ where λ^2 resembles the MSD. Now, if ℓ_0 is the smallest distance between last visited target and searcher's position in the step after finding that target, in the case of non-destructive foraging, the average number of flights would be $N_n = \frac{(\lambda-\ell_0)(\ell_0)}{2D}$ because the previous target can be revisited and the forager would be highly likely ℓ_0 away from the target [2]. Therefore, the average number of steps between two successive targets in the case of non-destructive foraging can be written as:

$$N_n \approx (\frac{\lambda}{r_v})^{\frac{\mu-1}{2}} \tag{2.8}$$

By plugging in the equations (Eq. (2.7)) and (Eq. (2.6)) into the efficiency equation (Eq. (3.3)), and differentiating with respect to the Lévy exponent, μ, it turns out there is no maximum for efficiency when in destructive foraging and efficiency increases as μ approaches 1[2]. For non-destructive foraging, using (Eq. (2.8)) in the efficiency equation, and taking the derivative with respect to μ results in a maximum where $\mu_{opt} = 2 - \delta$ where $\delta \approx \frac{1}{[ln(\lambda/r_v)]^2}$. Given the fact that $\lambda \gg r_v$, δ will be very small, optimal value for the Lévy exponent will be equal to 2 (Fig. 2.2 c) which means a combination of localized searches and delocalized searches is the optimal foraging strategy.

In addition to the mean-field approach above, the number of steps between successive targets can be thought as average time spend between absorbing boundaries where one boundary is at $X = 0$ and the other boundary is at $X = \lambda$, and it was calculated by using fractional differential equations and finding the probability of being

absorbed at the n_{th} flight [35, 36]. The average number flights then will be written as the following general form:

$$N_{\mu}\left(x_0\right) = \frac{\sin[\pi(\mu-1)/2]}{\pi(\mu-1)/2}\left[\frac{\left(\lambda - x_0\right)x_0}{\ell_0^2}\right]^{(\mu-1)/2} \tag{2.9}$$

Where x_0 is the initial position and ℓ_0 is the minimum step length which would be equal to r_v. For destructive foraging, the searcher will start in the middle, so $x_0 = \frac{\lambda}{2}$. For non-destructive foraging, due to the fact that the target grows back, the forager will start closest to the previous target, and $x_0 = \frac{\lambda}{2}$. For each case, destructive and non-destructive, N_d and N_n can be calculated by plugging in the respective initial position into (Eq. (2.9)) which results to equations similar to (Eq. (2.8)) and (Eq. (2.7)).

2.2 Concepts for chapter §4

2.2.1 Collective behavior

Organisms across scales come togethers and form groups to complete different task [3]. Schools of fish [37], flocks of birds [38], bacteria swarms [39], swarms of flies[40, 41], and cellular clusters [20] are examples of collective motion where these living organisms act as a single unit that is very different than the individuals behavior (Fig. 2.3). In such systems, individuals are self-propelled, and they consume energy to move. Moreover, they interact with each other locally, and form a group with length scale much larger than each individual [3]. The interactions between the individuals can be simple such as repulsion and attraction, or can be more complicated and a combination of many simple interactions. In collective behavior, an individual's behavior can be dominated by the influence from the others. Eventually, such systems show ordering patterns as the individuals constantly change their behavior to follow the group [42].

Physicists treat such groups or flocks as a system of particles, and apply statistical physics methods such as theories on scaling and computer simulations on the systems with collective behavior. The most common approach is to treat the individuals as similar particles, that move with a nearly constant absolute velocity. The particles are capable of changing their direction of motion based on their neighbors direction

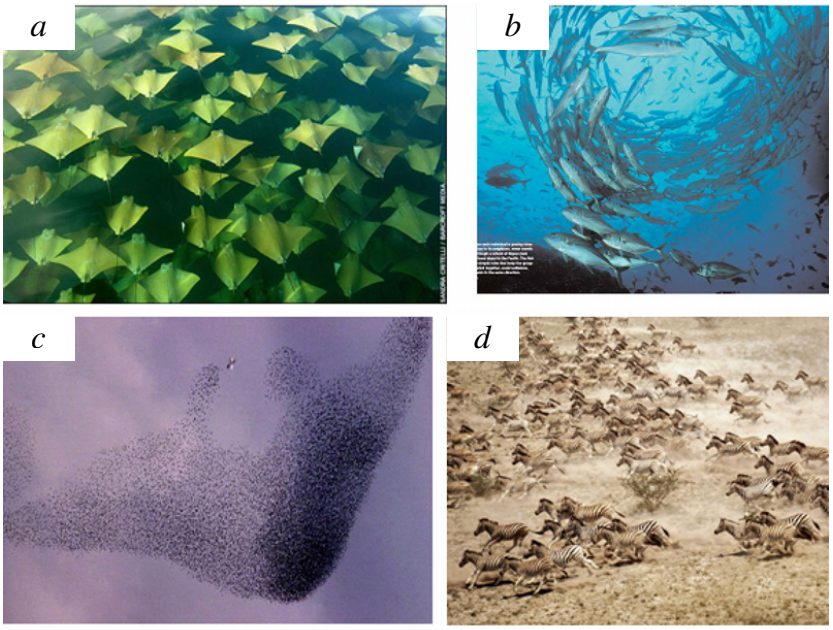

Figure 2.3: Examples of collective behavior among organisms (a) array of golden rays, (b) school of fish (c) flock of starlings (d) a herd of zebra. Figures are from [3].

through an alignment interaction. They may also experience additional forces from the neighbors as mentioned above, and they are also subject to a noise [3, 43].

Vicsek showed that allowing the agents to interact with each other by aligning their direction of motion with the neighboring agents, results in a noise driven phase transition from an ordered to a disordered state as noise increases [43]. The phase transition can be defined by an order parameter. Considering each particle to move with some velocity $\overrightarrow{v_i}$, having N particles, and v_0 being the average absolute velocity of each particle, the order parameter is defined as:

$$\phi = \frac{1}{Nv_0}|\sum_{i=1}^{N}\overrightarrow{v_i}| \tag{2.10}$$

If the particles move in different directions and motion is disordered, the order parameter would be 0 and if particles move in the same direction the order parameter sums up 1. This noise driven phase transition inspired physicists to explore collective behavior and find new phases such as rotational, disordered, jamming and fully ordered phases[3]. Two main approaches to study collective behavior are agent-based modeling, and hydrodynamic modeling. In agent-based models, interactions with other

agents are calculated individually and positions are updated according to the interactions over time. The second approach is hydrodynamic modeling where local density and polarization are defined through continuum equations which can be numerically integrated to model the behavior of collective motion with time[44]. In Chapter §4, I use an agent-based model to study the collective foraging of cells.

2.2.2 Collective cell migration

Collective cell migration plays an important role in a variety of physiological processes such as wound healing and repair [45, 46], the morphogenesis in development of embryos [47, 48], and tumor formation and cancer metastasis [45, 49, 50]. In collective migration of cells, many cells come together and move in the same direction at a roughly similar speed (Fig. 2.4). Similar to collective behavior in many other systems, cells adjust their response to the environment to ensure they follow the group which results in a more efficient migration compared to individual cells. Single cells typically have higher velocity. However, they change direction more frequently and their motion is not persistent [51].

Collective behavior of cells requires chemical and physical communication between individual cells. It has been shown that direct cell to cell contact as well as the internal molecular mechanisms that control cells polarization play important rolls in collective motion of cells [51]. Single cells form protrusions such as filopodia and lamellipodia and interact with extracellular matrix which allows them to move forward [52]. The same mechanism happens in a group of cells that move together. However, cells in a group are affected by cell to cell contact, and the distribution of protrusions will be altered. The cells in the front of the clusters are considered leader cells, and the ones in back are considered follower cells.The leader cells make more protrusions, and are more exposed to external signals such as chemoattractants. Follower cells, are not able to form such protrusions due to cell to cell contacts, and they rely on cellular interactions and alignment with other cells[51].

Leader cells are located at the front of the cluster. However, follower cells can relocate and become new leader cells. This relocation could be due to the response to external cues such as soluble factors like chemotaxis and interactions with neighboring cells[46]. For example. signaling through growth factors or chemokines can result in cell polarization and protrusions which determines the leader cells[53, 54, 55, 56].

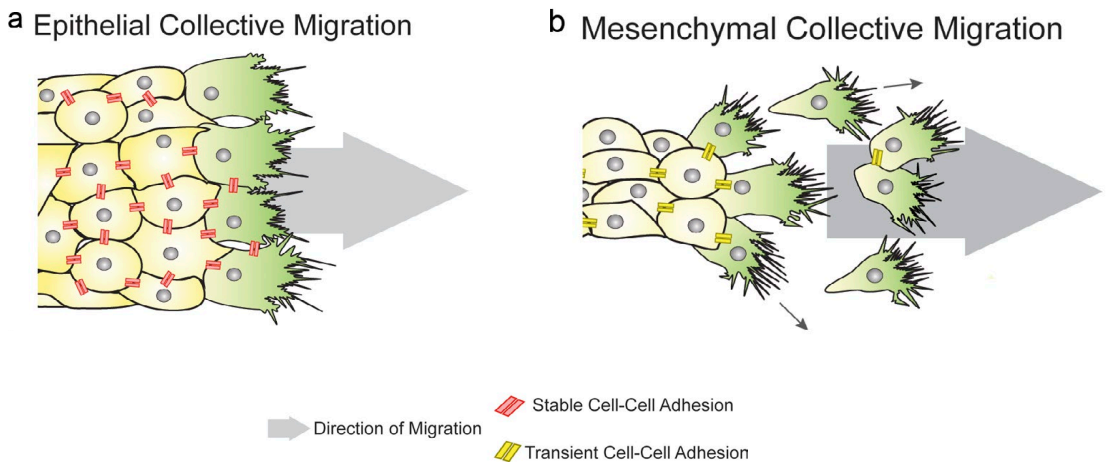

Figure 2.4: Examples of collective cell migration. (a) Epithelial cells migrating as a cohesive group, maintaining cell-cell adhesions. Leader cells form protrusions oriented in the direction of migration. (b) Mesenchymal cells migrate in the same direction collectively. However, they form loose and transient cell-cell connections. Images are from[4].

Moreover, intracellular signaling can control gene expressions and define the characteristics of leader cells which comes from chemotactic factors. In such cases, cells that are more responsive to chemical signals become the leader cells[57, 58, 59]. As the group of cells migrate, position of the leader cells can be challenged, and they can be exchanged with follower cells and switch roles. Follower cells might become new leaders to participate in chemotaxis and gradient sensing [60, 61]. Therefore, both leader cells and follower cells play an important roll in collective migration and robust chemotaxis[51].

2.2.3 Collective behavior of malignant lymphocytes

Malignant B & T lymphocytes exposed to $CCL19$ or $CXCL12$ migrate individually and form clusters. These clusters migrate collectively and they have a higher chemotactic sensitivity than individual cells which move more randomly. They were placed in a variety of chemical gradient from 0 $ng\ ml^{-1}\ mm^{-1}$ to 1000 $ng\ ml^{-1}\ mm^{-1}$. In shallow to moderate gradients both individual cells and clusters sense the gradient and

travel toward higher concentrations. However, in high gradients ($\geq 500\, ng\, ml^{-1}\, mm^{-1}$), individual cells are subject to chemo-repulsion, and their receptors get saturated and they go to lower concentrations of chemokine, while clusters move in a very directional manner (Fig. 2.5) [62].

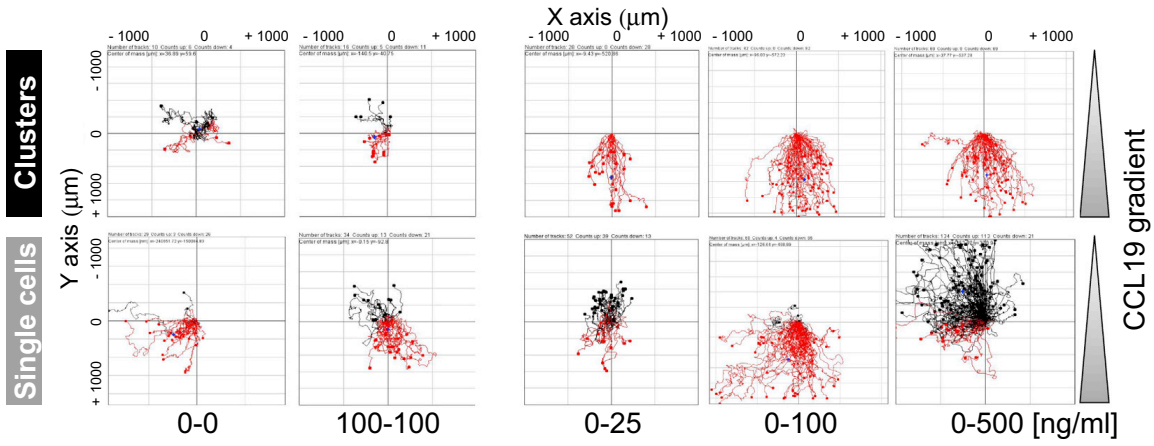

Figure 2.5: Examples of trajectories of clusters of malignant lymphocytes (top panel), and individual cells (bottom panel) in various gradient from 0 to 500 ng/ml. In high gradients individual cells go towards lower concentrations of CCL19 while clusters are still able to climb up the gradient. Images are from[62].

To understand the reason behind clusters robust chemotaxis. they tracked the cell nuclei and extracted the velocity fields in frames $15s$ apart from each other. It was shown that these clusters exhibit three novel collective phases. They observed a running phase where cells are aligned together and move in the same direction. There is also a rotating phase where cells rotate around the clusters center, and finally a random phase where cells velocities are uncorrelated. It was also shown that cells shuffle between the rim and core of the cluster. Furthermore, they observed the leader cells receptor, $CCR7$, internalizes, and the leader cells lose their polarity and protrusions. It is speculated that the rotating phase and the exchanges help the clusters to chemotax. When the leader cells receptors get saturated, a new cell can become the leader by the exchanges. Moreover, the cluster can rotate and a cell with fresh receptors can move to the leading edge to take the leader roll. In chapter §4,

by analyzing the trajectories of the clusters, I show the internal dynamics of the cells that allow the clusters to migrate collectively.

Chapter 3

Optimal foraging strategies for territorial competitors

3.1 Introduction

Optimal Foraging Theory (OFT) considers that animals aim to maximize a currency such as net caloric gain per unit time, subject to constraints that could be physiological or environmental. For fixed constraints, therefore, OFT would predict that organisms would adopt the most efficient search strategy [2].

The search strategy can be guided by external cues such as visual, auditory or olfactory stimuli or even previous memories, or it might not be directed in which case it becomes a random search process. Thus, when the locations of resources are not known *a priori* and there are no directional cues, a natural question that arises is whether organisms can optimize a completely stochastic search [1].

In such a situation, in the case of sparse targets, many animals such as albatross exhibit foraging patterns where distances traveled are drawn from a heavy tailed Lévy distribution, $P(l_j) \sim l_j^{-\mu}$, where $1 < \mu \leq 3$ [2, 8], with the direction of movement chosen from a uniformly distributed angle. When $\mu < 1$ the motion is considered ballistic and with $\mu > 3$, Brownian motion. Similar behavior was observed in other organisms such as jackals [6], bacteria within a swarm [63], T-cells [64], and spider monkeys [7].

It has been shown theoretically that, in the case of sparse and random regenerative targets, the search efficiency is optimized around $\mu = 2$, where the efficiency, (η), is

defined as the ratio of total number of targets found, to the total distance traveled [2]. This matches the behavior of several foraging animals [2], lending validity to the model. Several extensions to this basic model have been studied, yielding insights into realistic foraging strategies. When resources become non-regenerative, for example, no optimal μ value is observed with efficiency decreasing as μ increases [1, 2, 18, 19]. Targets can also display various types of spatial distributions such as patchy, or normally distributed, and can be motile like plants or stationary like animals. For cases of low target density, when targets are non-regenerative and Lévy distributed with exponent close to 3, an optimal Lévy exponent for the searcher is observed around $\mu \approx 2$ [65]. Furthermore, when targets are distributed in patches or with a Lévy distribution [66] with high density, less super-diffusive strategies, $(2 < \mu < 3)$, perform nearly close to $\mu = 2$ [66]. The topography of the environment can also have an effect on search efficiency. For example, when the environment has a concave porous topography [67], the search is optimized for destructive foraging (non-regenerative resources) around $\mu = 2.4$.

While solitary foragers have been extensively studied, in many natural settings, multiple organisms cooperate or compete with each other for resources. Studies have shown that Brownian searchers with with even rudimentary purely repulsive interactions can minimize their mean first-passage time (MFPT) to targets with optimal, intermediate values of the interaction strength [68]. Furthermore, search times are minimized for both Brownian and Lévy searchers when the range of cooperation is optimized, but Lévy strategies can be faster [69]. Studies on the effects of communication on the foraging patterns of Mongolian gazelles showed that communication over intermediate length scales leads to a faster search and minimizes the MFPT to targets [70]. Mixtures of strategies have also been shown to help cooperative foraging. For example, the search efficiency of a group of foragers, who can either search independently or by following others who find target patches, is maximized for a mixture of the two strategies. If searchers only follow other successful individuals, target patches might become depleted before they arrive at the site [71].

While many animals cooperate, many others such as coots [72], and tigers [73] are territorial animals, and much less is known about the effects of competition and specifically territorial competition on foraging. Here, I develop stochastic agent-based simulations that model foraging with competitive interactions mediated by

territorialism. I explore different types of strategies that can optimize group foraging in these cases. In our simulations, each identical agent has a territory with a fixed size, r_t, around itself which is not accessible by the other searchers. I study the effect on foraging efficiency of varying the territory size and the number of foragers. We do this with two different protocols. In the first protocol, I study terrestrial animals, who are not able to cross a competitor's territory, such that they are forced to stop at the intersection of the excluded region and their own path. In the second protocol, I study aerial animals. In this case, foragers are allowed to cross others' territories, but they cannot land in it. I study the difference between the efficiency of a group of competitors and a single searcher performing Lévy flights of varying index μ. I look at varying the density of agents and size of their territory, r_t, and how it affects the search efficiency, and thus the search strategy of territorial competitors while performing Lévy flights. I also compute the variance in efficiency among multiple competing foragers as a function of the system parameters with a view to shedding light on optimizing searches in situations where minimizing variability is an important factor in addition to maximizing efficiency.

3.2 Model and simulation

Here I describe our agent-based model that I used to study foraging in a group of territorial competitors. Each individual agent performs a random walk consisting of a series of steps in random directions and has a perceptive range, r_v, within which it can detect resources. The step-lengths are drawn from a heavy-tailed Lévy distribution that is bounded:

$$p(l_j) = \frac{\mu - 1}{l_0^{1-\mu} - l_{max}^{1-\mu}} \left(l_j\right)^{-\mu} \tag{3.1}$$

The Lévy exponent is within the range of $\mu \in (1, 3]$. The smallest step-length I allow the forager to take is $l_0 = r_v$ because steps smaller than the vision radius, r_v, will not be beneficial. The maximum step-length, l_{max}, is equal to L, the size of our system. This is due to the fact that steps larger than the size of a landscape, L, are not realistic and unbounded displacements, or infinite step-lengths, are naturally forbidden [66]. The mean free path between two successive targets is defined as:

$$\lambda = (2r_v\rho)^{-1}, \tag{3.2}$$

where ρ is the target density (total number of targets over total area). The efficiency of the search, (η), is the ratio of total target found (N_{total}) over total distance traveled (L_{total}), or the inverse of average flight length $< l >$ times the average number of flights (N) between two successive targets:

$$\eta = \frac{N_{total}}{L_{total}} = \frac{1}{< l > N} \tag{3.3}$$

The Lévy flight foraging procedure for a single agent with targets works as follows [2]:

1. At each time step, if there are multiple targets within the foragers perceptive range, r_v, the agent goes to the nearest target.

2. If there are no targets nearby, the forager picks a random flight length, l_j, from the Lévy distribution (Eq. (3.1)), and a random uniformly distributed angle between 0 to 2π, and starts the next flight.

3. The forager is constantly looking for targets within its vision radius, r_v, while it is taking the steps along its way.

4. If a target site is within r_v, the forager goes to it. Otherwise, it completes that flight path, l_j, and repeats steps 1 and 2.

For a solitary forager, the average number of flights between two successive targets, N, depends on whether the search is destructive or non-destructive (non-regenerative or regenerative resources). For destructive searches, $N_d \approx (\frac{\lambda}{r_v})^{\mu-1}$, and $N_n \approx (\frac{\lambda}{r_v})^{\frac{\mu-1}{2}}$ for non-destructive searches [2, 35]. For a solitary agent, the efficiency is maximized as a function of Lévy index μ, with value $\mu = 2$, [2].

To model multi-agent foraging I consider N_f foragers which are randomly placed in a two dimensional box of size L with periodic boundary conditions. The periodicity is applied to the movement of the foragers, as well as the regeneration of the targets. Targets are distributed randomly, from $[0, L]$ in x and y, and they are regenerative such that at each time step, I have a fixed number of targets. Each forager has a territory with radius r_t around itself which is not accessible by the other foragers. Individual foragers perform flights according to the Lévy flight foraging procedure specified above with modifications due to interactions detailed below. At each step,

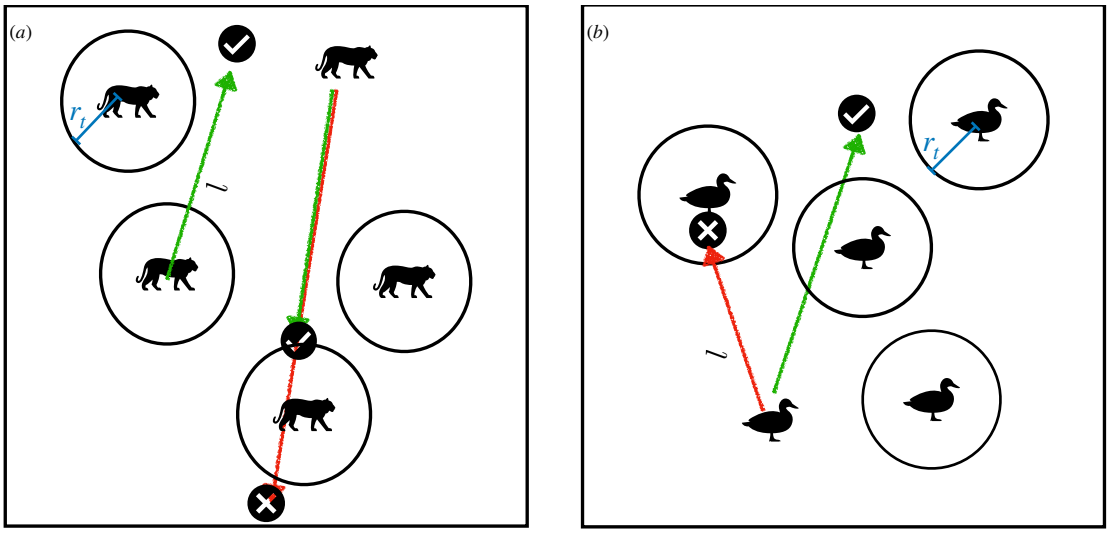

Figure 3.1: Schematic illustration of foraging protocols. (a) Terrestrial animals are not allowed to cross other foragers' territories. (b) Aerial animals are allowed to cross other foragers' territories, but cannot land in it.

foragers perform their flights in a random order to avoid bias. I define two different protocols for our foragers to model terrestrial and aerial animals. In our first protocol, terrestrial, foragers are not able to cross another forager's territory, and they are forced to stop at the intersection of the excluded region and their path. In our second protocol, aerial, foragers are allowed to cross other foragers' excluded region but they cannot land in it. The foraging pattern, for the first protocol (Fig. 3.1 a), terrestrial, is then as follows:

1. At each step, a random order of foragers is chosen. When one forager finishes the following steps, the next forager starts. By the end of the step, all agents have performed one flight.

2. The chosen forager picks a random flight length, l_j, from the Lévy distribution (Eq.(2.1)), and a uniformly distributed random angle.

3. The forager starts moving, and uses Lévy flight foraging procedure to find targets [2].

4. If the forager's path intersects with other territories, the forager stops at the intersection of its path and the excluded region.

5. If a target is inside of the other foragers territory, the forager will skip that target, to remain consistent with the given protocol.

6. Steps 1 to 5 are repeated until the maximum number of steps of the simulation is reached.

The foraging pattern for the second protocol (Fig. 3.1 b), aerial, is similar to the terrestrial protocol, except for step (4). In this case, if the end point of the forager's flight is inside of another forager's excluded region, step (2) needs to be repeated. I note that the first protocol is meant to represent terrestrial animals because they can sense competitor territories and I assume that they stop the moment that they hit the periphery of an excluded region. Similarly, the second protocol represents aerial animals, since the agents are able to see the end point of their flight at the beginning, and choose another random flight path to avoid landing inside a competitor's territory.

3.3 Results

The interaction between the foragers affects the search efficiency, and potentially changes the optimal foraging strategy by changing the encounter rate. Two factors influence this rate, the number of foragers, N_f, and radius of their territory, r_t. By increasing the number of foragers, as well as the radius of the territory, the encounter rate between foragers will increase. In what follows, I study cases with two, four and eight agents with territory radius $10 < r_t < 100$. Unless otherwise stated, the simulation box size, number of targets and vision radius are fixed as $L = 500$, $N_{targets} = 25$, $r_v = 1$.

In the terrestrial case (Fig. 3.2 T_{1-3}), we notice that increasing the radius of territory, r_t, and the number of agents, N_f, leads to lower efficiencies (Fig. 3.3) overall. However, the optimal Lévy index μ is still approximately equal to 2 (Fig. 3.3). To investigate this further, we look at the inverse of average number of flights ($\frac{1}{N}$) and the inverse of average flight length ($\frac{1}{<l>}$) as a function of radius of territory, r_t, for different values of μ since the efficiency is defined as $\eta = \frac{N_{total}}{L_{total}} = \frac{1}{<l>N}$. We observe that steps become truncated, so that the inverse average step length increases as a function of r_t (Fig. 3.4 a). At the same time, fewer targets are accessible to each forager because they are enclosed by other agents' territories. Therefore, the inverse of number of steps between two successive targets decreases as a function of

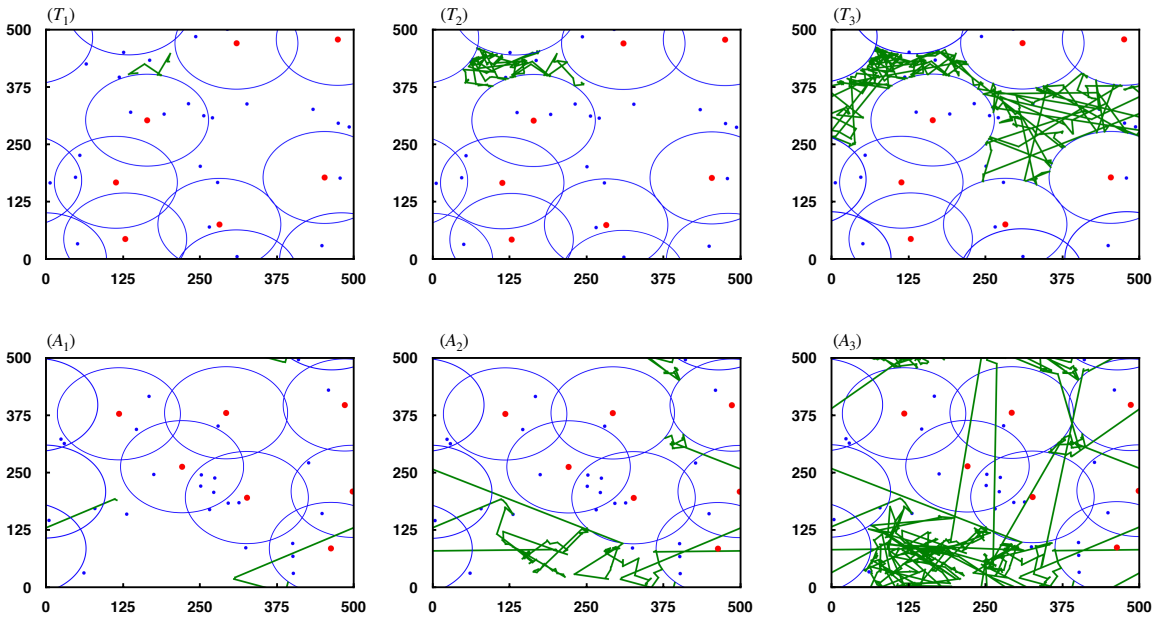

Figure 3.2: In the top row, T_{1-3}, are snapshots of simulations (after 10, 150 and 500 flights respectively) for the terrestrial case, with $N_f = 8$ and $r_t = 100$, where the green path represents the foraging pattern for 1 forager. The bottom row, A_{1-3}, shows snapshots of simulations (after 10, 150 and 500 flights respectively) for the aerial case, with $N_f = 8$ and $r_t = 100$. $\mu = 2$ in both cases. For visualization purposes, the other 7 agents are stationary in this set of simulations.

r_t for different values of μ (Fig. 3.4 a inset). The relative reduction in the number of targets found is however larger than the reduction in the distance traveled (Fig. 3.4 b) leading to net decrease in efficiency with increasing r_t for μ greater than about 1.4. However, counter-intuitively, we note that the efficiency is *higher* for larger r_t and larger number of foragers when μ is smaller than 1.4 (Fig. 3.3). The reason for this is that, for small μ and small r_t values, very long jumps are more likely to occur. So, the agents end up taking long jumps without finding as many targets. Therefore, the natural truncation in the step-lengths for larger r_t values is in fact beneficial for the agents and it prevents them from traveling long distances without finding resources. This is also reflected in the fact that the relative reduction in the number of targets found is *smaller* than the reduction in the distance traveled for $\mu < 1.4$ (Fig. 3.4 b). Thus, territorial competition can be beneficial in the limit of low μ, de-localized search strategies.

When we compare the aerial (Fig. 3.2 A_{1-3}) and the terrestrial (Fig. 3.2 T_{1-3})

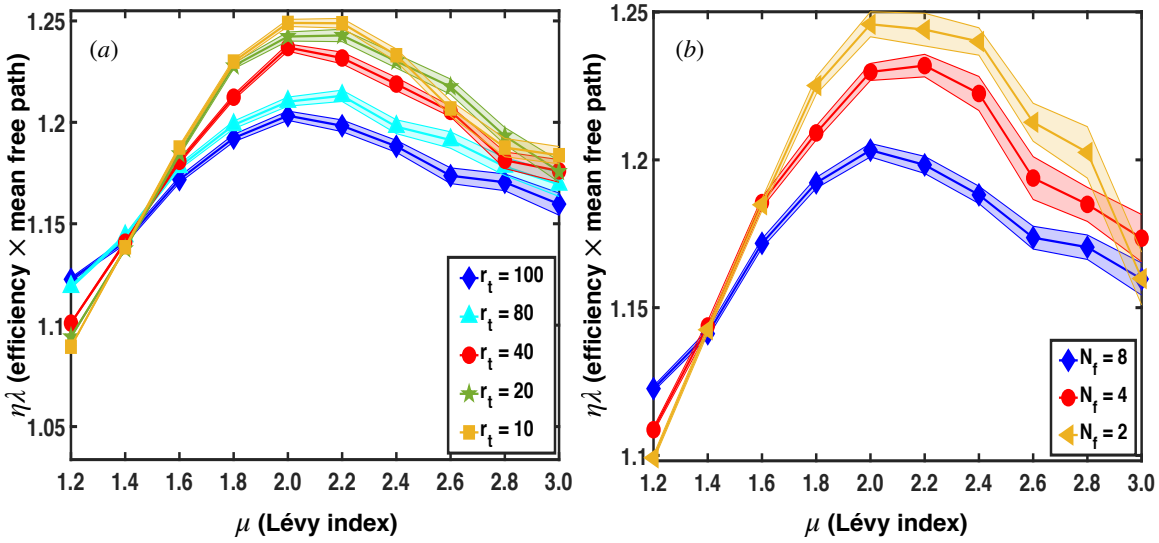

Figure 3.3: (a) Efficiency η versus μ for eight agents ($N_f = 8$), and different radii of territory ($r_t = 10, 20, 40, 80, 100$). (b) Efficiency η versus μ for $r_t = 100$ and different number of agents ($N_f = 2, 4, 8$). The spread is the standard error of the mean and the solid lines are the average efficiencies.

cases, we notice that, as for the terrestrial case, the efficiency in the aerial case decreases when the number of foragers, and radii of territory increases (Fig. 3.5). We also observe a lower efficiency for the aerial case compared to the terrestrial case that is more pronounced for lower values of μ. The average flight length in the terrestrial case is lower than the aerial case, since in the aerial case, agents are still allowed to take longer jumps. Since the steps still come from the same Lévy distribution, bigger flight lengths can occur (Fig. 3.6 a). However, in the terrestrial case, $< l >$ decreases by increasing r_t because foragers are forced to stop if their path intersects with other territories. The total number of targets found in both cases decreases by increasing r_t, since targets in other foragers territories, are not accessible to all of the foragers. However, the number of targets found does not significantly increase with decreasing μ, compared to the terrestrial case which is shown in (Fig. 3.6 a inset). This can also be seen in the ratio of total targets found in the aerial case to the terrestrial case as well as the ratio of the average flight lengths in the two cases, plotted as a function of μ in (Fig. 3.6 b). We see immediately that the average flight length, $< l >$, ratio is significantly higher than the ratio of targets of found N_{total} for smaller values of

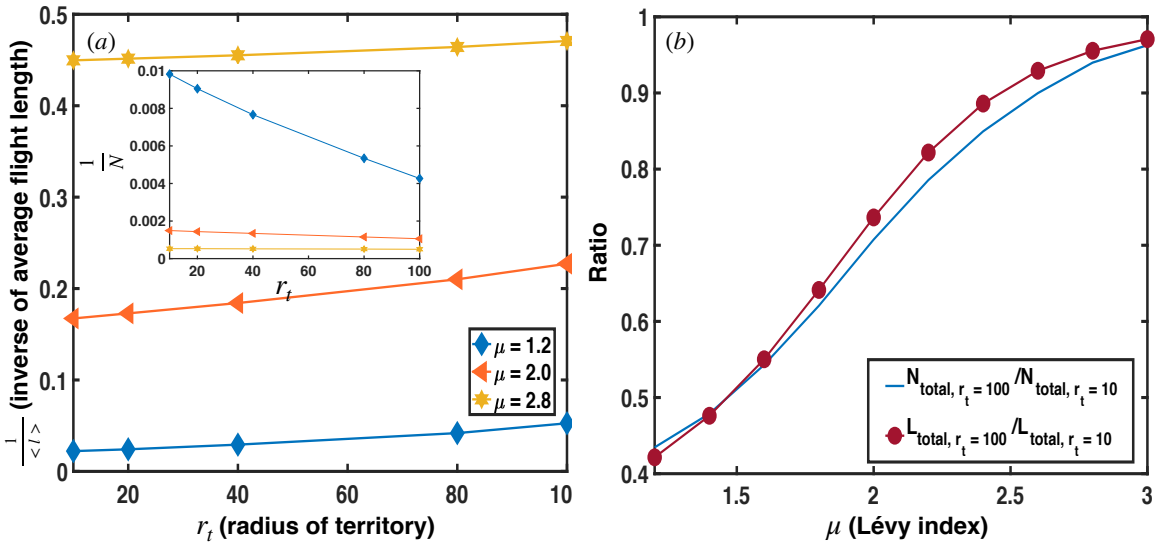

Figure 3.4: (a) Inverse of average flight lengths, $\frac{1}{<l>}$, as a function of r_t. Inset is the inverse of the average number of flights between two successive targets, $\frac{1}{N}$. Both plots are for different values of Lévy index $\mu = 1.2, 2.0, 2.8$. (b) Ratio of total targets found (blue) and total distance traveled (red) for $r_t = 100$ and $r_t = 10$.$N_f = 8$.

μ (Fig. 3.6 b). This results in a greater suppression of the efficiency in the aerial case for small values of μ.

While we have so far considered the mean efficiency of the population, we now consider a measure of the variance by computing the standard deviation of the efficiencies among agents. We also look at this standard deviation for many solitary foragers with different starting points. We consistently observe a higher standard deviation for higher μ values even after traveling long distances (Fig. 3.7 a), and the deviations are of comparable magnitude for the terrestrial, aerial and solitary searchers. Therefore, there is no significant difference between territorial searchers and solitary searchers in terms of the variance of the efficiency among foragers. We note that, though the standard deviation will eventually vanish after long enough times, it is important to consider variance among individuals at intermediate times that could be of biological relevance, such as seasons or reproductive intervals. This standard deviation, in fact, increases monotonically as μ increases in all cases (Fig. 3.7 b). This indicates that foraging strategies with higher μ values, or shorter step lengths may lead either to a highly efficient search or a search with an efficiency well

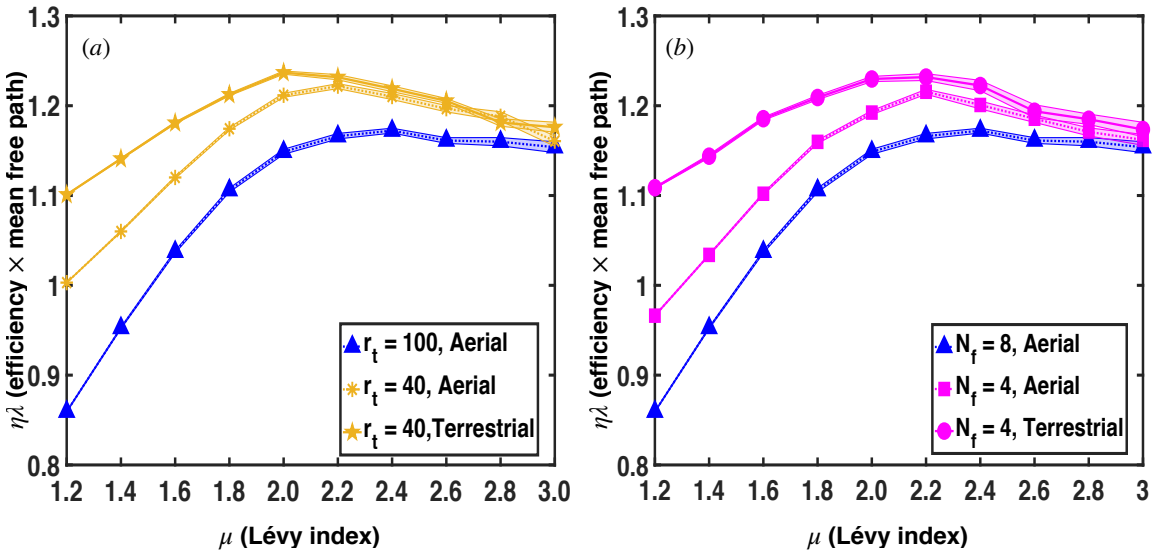

Figure 3.5: (a) Efficiency for the aerial case ($r_t = 40, 100$) and the terrestrial case ($r_t = 40$) for $N_f = 8$. (b) Efficiency for the aerial case ($N_f = 4, 8$) and the terrestrial case ($N_f = 4$) for $r_t = 100$.

below the average efficiency of the population. For a search with smaller Lévy index and larger flight lengths, on the other hand, the variance is small, and all the agents perform a search with efficiency close to the average. We can rationalize this by considering that at higher μ values, due to the smaller step sizes, less space is sampled within a certain time, and so if an agent is in part of a space which has more (or less) resources, it will have a more (or less) efficient search. For smaller μ values, longer flights are dominant, and chances of visiting different spots of the landscape within the relevant time will be higher. Therefore, the searcher is able to better sample the entire space, resulting in a smaller variance.

Finally, we look at the efficiency in the destructive case, where targets will not be able to grow back after they are found. In the terrestrial case, the behavior is similar to a single searcher [1] with no optimal value for μ (Fig. 3.8 a), and the efficiency decreasing as μ increases. We also see that increasing territories results in slightly suppressed efficiencies. In the aerial case, however, we see a peak in the efficiency as a function of μ especially for higher r_t, indicating the existence of an optimal strategy for destructive foraging in this case. This peak arises from the same effect in the non-destructive aerial search where the efficiency is suppressed for smaller μ values and

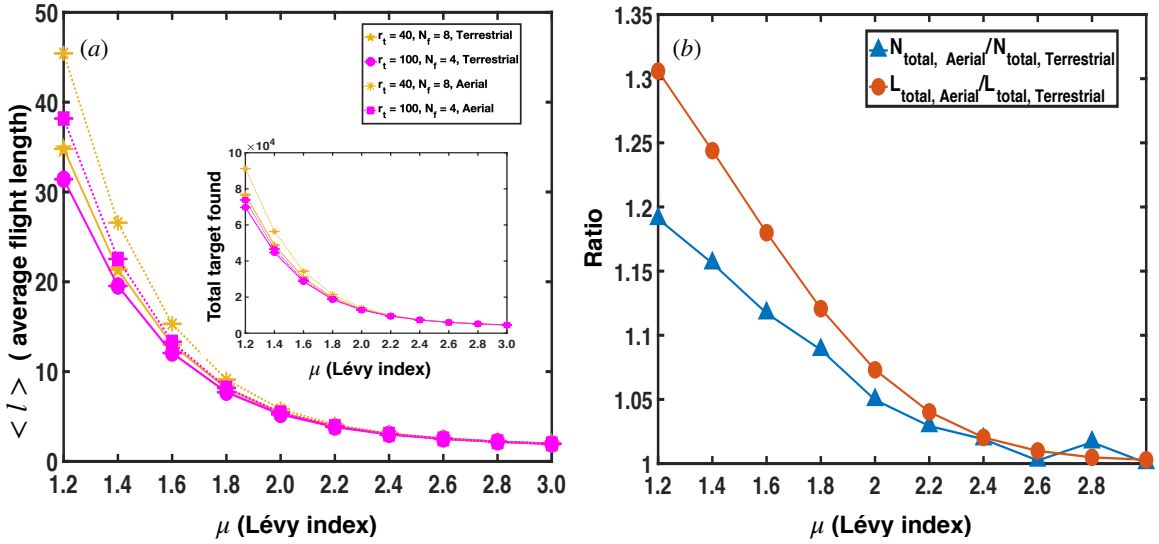

Figure 3.6: (a) Average flight length for the terrestrial (solid lines) and aerial (dashed lines) case. Inset is total target found for the terrestrial(solid lines) and aerial(dashed lines) case. (b) Ratio of total targets found (blue) and total distance traveled (red) between the aerial and the terrestrial cases for $r_t = 40$ and $N_f = 8$.

higher r_t, since longer jumps are allowed, but agents cannot access the targets (Fig. 3.8 b). In this case, this suppression creates a slight peak in the efficiency around $\mu = 1.6$ for higher r_t values (Fig. 3.8 b). We note that the optimum shifts to the left and becomes less pronounced for smaller r_t indicating an optimum that is tunable by territory size.

3.4 Discussion

It has been established that many solitary foragers such as goats [74] and spider monkeys[7], as well as bacteria[63], and cancerous cells perform Lévy flight type search patterns while looking for sparse, randomly located resources. While the actual statistics of the searches are debated and myriad factors including memory, topography, spatial and temporal distribution of resources can affect the optimal strategy [75, 76, 77], it is clear that searches do contain steps from long-tailed distributions and optimization principles are at work. The analysis of simple, minimal models have provided rigorous, quantitative frameworks to analyze such behavorial patterns and

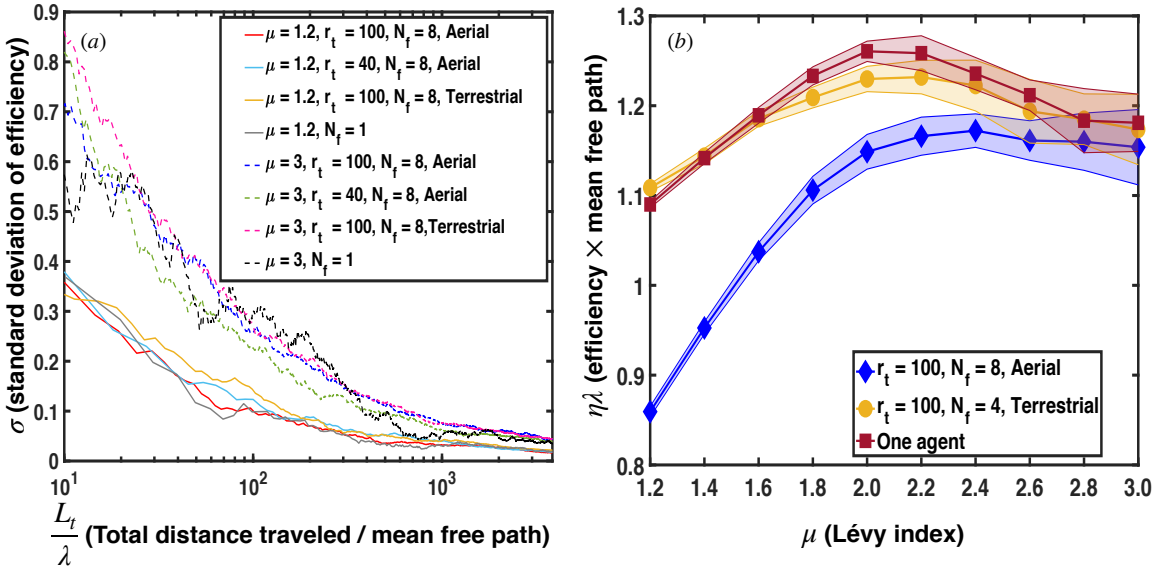

Figure 3.7: (a) The standard deviation of the efficiency over distance traveled for the terrestrial, aerial and single forager cases. (b) Efficiency as a function of μ for the aerial, terrestrial and single forager cases. The shaded region around the mean efficiency is the standard deviation measured after $N = 10^7$ flights.

uncover potential reasons for observed strategies. In this spirit, we have introduced a minimal model of group foraging with territorial competition to understand the effect of territorialism on foraging strategy.

For a single searcher, the efficiency of search is maximized when a combination of localized and non-localized steps are taken. In the case of sparse targets, the most beneficial search strategies observed are Lévy flights with $\mu \approx 2$ [2]. We showed that, in the presence of competition, strategies maximizing the efficiency are similar to those for single searchers and the optimal Lévy exponent, μ, is still approximately 2. However, in both terrestrial and aerial animals, the efficiency of the search generally decreases when the number of agents and the size of territory increases, i.e. increasing competition leads to lower overall efficiency for the group. However, for μ values close to 1 in the terrestrial case, larger territories, limiting the motion of foragers, are beneficial and increase the search efficiency because they cause a truncation in foragers step lengths. This truncation prevents foragers from traveling long distances without finding targets. Thus an increase in territorial competition can increase the efficiency of the group.

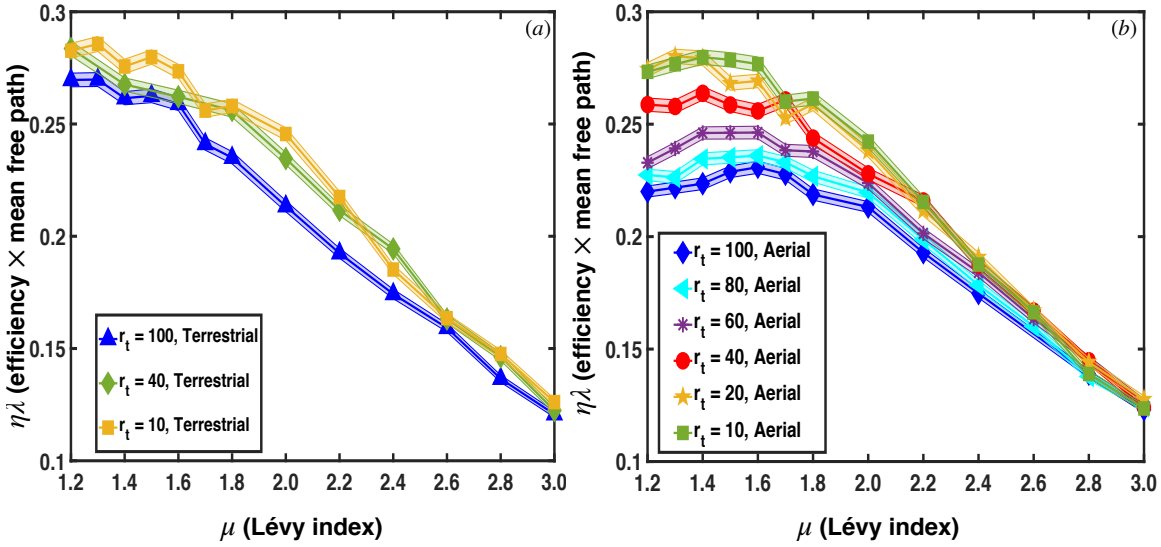

Figure 3.8: Non-regenerative search efficiency as a function of μ. The spread is the standard error of the mean and the solid lines are the average efficiencies. (a) For the terrestrial case, we observe no optimal Lévy index μ and the behavior is similar to a single forager. (b) For the aerial case, we observe a tunable optimum in efficiency as r_t increases. $N_f = 8$ in (a) and (b).

In addition to looking at the mean efficiency of the group, we also focused on the variance in efficiencies among individuals which is potentially of biological significance in contexts where optimizing the lowest efficiencies in a group might be important. We found that the variance among the efficiencies of individual foragers in a group was similar to the variance in the efficiencies of many solitary foragers. For small Lévy exponent, μ, values the variance is small, and for large μ values the variance increases. This suggests individuals and groups have two limiting options: *(i)* Performing searches with $\mu \approx 2$ which optimizes the mean efficiency of the population at the cost a higher variance. *(ii)* Performing nearly ballistic searches which decreases the variance at the cost of decreased mean efficiency. Depending on the relative priorities of the foragers, they can choose strategies that interpolate between these two options. If maximizing the efficiency of the population is the goal, they can perform searches with larger μ values close to the optimal mean efficiency. If minimizing the variance is the goal, searching with smaller μ values is more beneficial. For instance, if a solitary forager or members of a population prefer to ensure a higher chance of

crossing a minimal efficiency (to avoid starvation) rather than try to maximize their mean efficiency, they could perform more long-ranged (smaller μ) searches. Interestingly such long-ranged searches may be more advantageous for territorial competitors than solitary foragers as their mean efficiency is also higher in this regime.

Finally, for destructive foraging, where targets do not regenerate after being consumed, the optimal search strategy for solitary foragers within the minimal model is purely ballistic. This can change when targets are distributed in patches or can occasionally evade capture or the landscape is porous and concave [65, 67, 75]. Here, we show that, for terrestrial foragers, similar to solitary foragers, the optimal strategy is still ballistic and the efficiency decreases as the size of territory or population of foragers increase. However, for aerial animals, where long jumps are allowed, and resources are limited, an optimum appears. For long-ranged searches, around $\mu = 1.2$, large territories limit the access to the targets by other foragers, and since crossing is allowed, agents end up taking very long jumps without finding any targets. This results in a suppression in the efficiency for small μ values and creates an optimum that depends on the territory size. This optimum shifts to the left from μ around 1.6 when radius of the territory is around 100, to μ around 1.4 when the radius is 60. The optimum eventually disappears for small territories since the targets become more accessible and the optimal strategy becomes ballistic.

Our work has shown that territorial competition can lead to improved efficiency of long-ranged searches and highlighted several factors that can shift the optimum strategy of foragers including selective pressure on minimizing the variance of the efficiency favoring lower μ or more long-ranged strategies, and aerial territorial competition leading to shorter-ranged optimal strategies. We hope that our results will help future work consider these additional factors quantitatively when analyzing foraging data from the field that show deviations from the simplest optimal strategies.

3.5 Appendix

3.5.1 Lévy Distribution

This section covers the Lévy distributions used for the random walk simulation in this chapter. The distribution used in the simulations has an upper bound which is equal to the size of our simulation box since infinitely long jumps are not realistic in nature. The lower bound is the vision radius of the agents because steps shorter than that are not beneficial. In this section both restricted upper bound and unrestricted are covered.

Lévy Distribution with restricted upper bound

- $\boxed{l_{max} = \text{Finite}}$

For Lévy distribution ($p(l) \sim l^{-\mu}$) [2], the normalized probability density function with respect to a lower, l_0, and upper bound, l_{max}, is the following:

$$p(l) = \frac{\mu - 1}{l_0^{1-\mu} - l_{max}^{1-\mu}} (l)^{-\mu} \tag{3.4}$$

where l_0 is the smallest step taken by the foragers which is equal to their perceptive range r_v, l_{max} is the largest step equal to the size of the box or the landscape and μ is the Lévy exponent such that $\mu \in (1, 3]$. The probability of a flight length, l, is given by the integration of equation (3.4), $P(l > l_0)$, from l to l_{max}.

$$\Pr(l > l_0) = \frac{1}{l_0^{1-\mu} - l_{max}^{1-\mu}} \left(l^{1-\mu} - l_{max}^{1-\mu}\right) \tag{3.5}$$

where $\Pr(l > l_0) \in [0, 1]$. Taking the inverse of equation (3.5) and solving for l, we find the flight length as,

$$l(\Pr) = \left[l_{max}^{1-\mu} + \Pr\left(l_0^{1-\mu} - l_{max}^{1-\mu}\right)\right]^{\frac{1}{1-\mu}} \tag{3.6}$$

Lévy Distribution with unrestricted upper bound

- $\boxed{l_{max} \to \infty}$

In the case of an unrestricted upper bound, we have the normalized probability

density function with a lower bound, l_0, and in the large limit shown as,

$$p(l) \approx \frac{\mu - 1}{l_0} \left(\frac{l}{l_0} \right)^{-\mu} \tag{3.7}$$

where l_0 is the smallest step taken by the foragers which is equal to their perceptive range r_v and μ is the Lévy exponent. The probability of a flight length, l, in this case is given by integrating equation (3.7), $P(l > l_0)$, from l to ∞.

$$\Pr(l > l_0) \approx \left(\frac{l}{l_0} \right)^{-\mu+1} \tag{3.8}$$

Taking the inverse of equation (3.8) and solving for l, we find the flight length as,

$$l(\Pr) \approx l_0 \times (\Pr)^{\frac{1}{1-\mu}} \tag{3.9}$$

3.5.2 Search algorithm and periodic boundary condition implementation

In this section, the method for the search algorithm with periodic boundary condition will be covered. I implemented two methods, one where I mirror the main simulation box around the box when a long jump is taken, and mirror the targets (Fig. 3.9), and the other method is cutting the long steps, moving the agent incrementally, modulating the steps within the main box, and dividing the main box into smaller squares to look for targets (Fig. 3.11). Both methods are explained below.

Slow search method using mirrors of the main simulation box

The basics of the search algorithm is similar to [2] which is explained in section §3.2. At each time a random angle θ with a uniform distribution is chosen. The step length, l, is chosen from a Lévy distribution with the method explained in §3.5.1 where the upper bound, l_{max}, is equal to the box side length L. Then, a box with dimension of $2rv \times l$, and two semi-circles at the ends of the box with radius r_v are drawn. First, I check to see whether there are any targets within the radius of r_v of the agent's location. If there is any, it goes directly to that target. If not, based on the direction of the step, and its length, the main simulation box will be mirrored in that direction (Fig. 3.9).

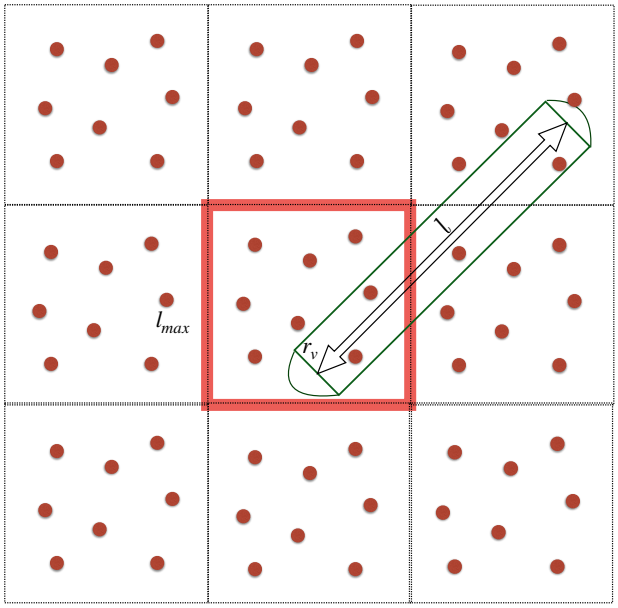

Figure 3.9: The simulation box is shown with solid black lines. The dashed boxes are the mirrors of the main box. When searcher takes a jump, a $2rv \times l$ box is drawn, and all the targets and their mirrors are checked to see if they are in this box. Two semi-circles at the beginning and the end of the final point of the steps is also checked with radius r_v. Then, the ones in the box and semi-circles are sorted based on their distance, and the forager goes to the one that is the closest to it. Targets are shown in red dots.

After that, we set the origin to be the lowest left corner of the box, and all of the targets positions within the main simulation box, and the mirrors will be checked to see if they are within the box and the semi-circle at the end. If there are targets within the box, we make a list and add those targets positions to the list. We then sort them based on their distance in an ascending order from the agents position, and the agent will be moved along the original path with angle θ and length l. When it gets within radial distance of r_v of the closest target, it deviates from the original path and goes to the target (Fig. 3.10).

Mirroring the targets at every time step, and checking the main and mirrored targets positions to see which ones are within the box is in the order of $\emptyset(N^2)$ time complexity. Moreover, when having the territories, targets need to be checked to see if they are in other agents territories, or in other words, if they are within distance r_t of any other agents. If they are within that distance, they need to be skipped and

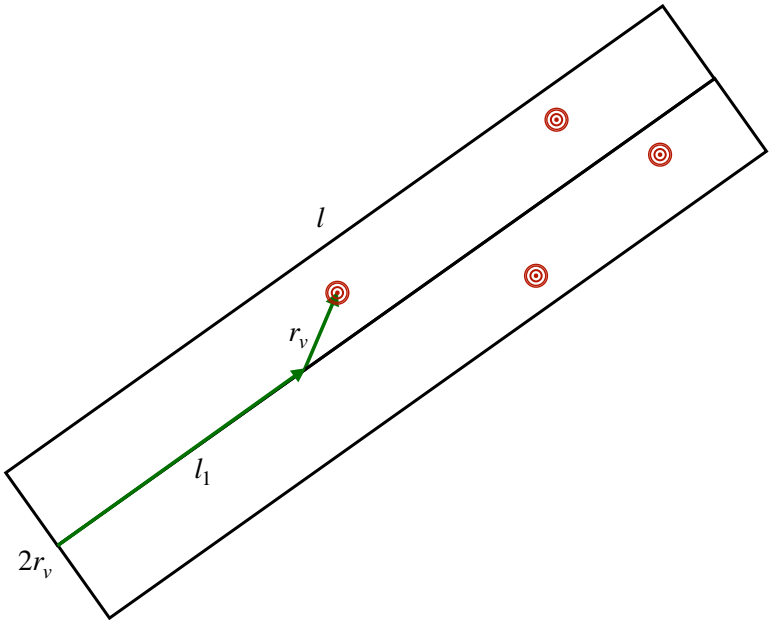

Figure 3.10: If there are targets detected within the box, the agent moves along the original path, until its distance with the closest target is r_v. Then, it deviates from the path and goes to the target. The agent in the figure moves the distance l_1 and distance r_v to get to the target.

the agent has to go to the next target. The time complexity of this algorithm results in slow simulations. Therefore, there was a need to come up with a faster method which is explained below (§3.5.2).

Fast search method using nearest squares to the agent

To make the search given the PBC faster, the box can be divided to smaller boxes with side length of $\widetilde{L}$ (Fig. 3.9) . Each box is labeled based on the coordinates of its lowest left corner. Then, if I floor each target's x and y position, I can figure out which box the target is located in by matching the floored position and box's labels or the coordinates of the lowest left corner. Thus, for each smaller box, there is a list which contains the targets that belong to that box. This only needs to be done once at the beginning of the simulation.

After that, the agent picks a random angle θ drawn from a uniform distribution, and a random step length, l, from the Lévy distribution with the method explained in (§3.5.1) where the upper bound, l_{max}, is equal to the box side length L. Next, the intersections of the agents path and the boundaries will be found, as well as the

positions where agent comes back to the box. Therefore, the path will be broken into smaller pieces shown in (Fig. 3.11). In each smaller piece, agent moves incrementally along the original path l, with increments equal to the length of the sides of small boxes, $\widetilde{L}$. In each increment, the four closest boxes to agents current location will be found by finding the closest corner to the agents location, and adding and subtracting $\widetilde{L}$ to that corner's coordinates. Given the four closest boxes, we will have a list of the targets within those boxes. Next, we draw a box similar to (Fig. 3.9) and (Fig. 3.10), but with length $\widetilde{L}$ instead of step length l, and check to see if there are any targets from the closest targets within this rectangle. If there is, the agent goes to the target. If no target is found within the increment, it will move again along the original path by $\widetilde{L}$. If it passes the boundary, it will come out from the other side of the box, and the same process will be repeated.

There is a trade off between choosing $\widetilde{L}$ very small, so there will be less targets in each box, and choosing $\widetilde{L}$ to be larger. Very small $\widetilde{L}$ results in less targets in each box. However, there will be more boxes to check, and smaller increments in which agents move along. Large $\widetilde{L}$ results in more targets in each box, but there will be less boxes to check for the agent in each time. I found the ideal $\widetilde{L}$ by going from $\widetilde{L} = 1$ to $\widetilde{L} = 250$ and measuring the time it took for the simulations to finish. The optimal time for the system shown in this chapter is $\widetilde{L} = 100$. This made the simulations to be at least $\times 10$ faster.

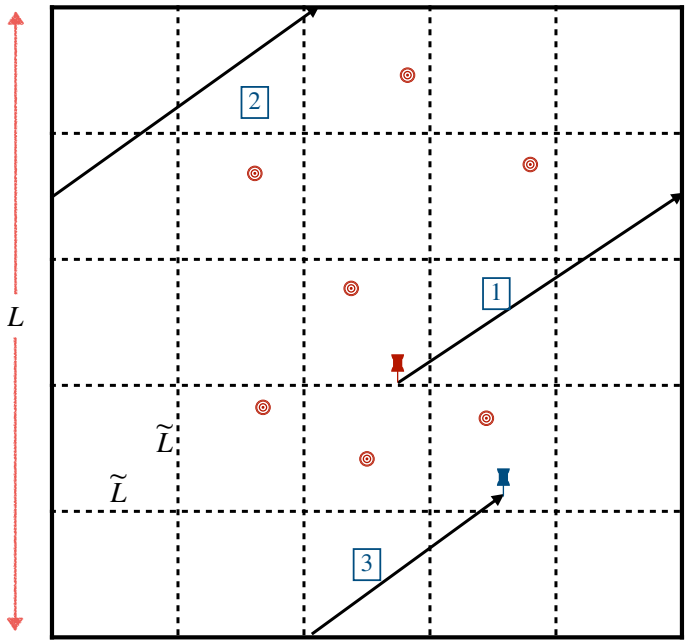

Figure 3.11: The simulation box is divided into smaller boxes with side lengths of $\widetilde{L}$. At each time, the only targets that need to be checked are the ones closest to the agents current position, or the box the agent currently is in. Each smaller box is labeled based on its lowest left corner. Targets locations are also found by taking the floor of their location. If the floor matches any of the boxes lowest left corner location, the target belongs to that small box. The agent is moved in smaller increments, and if any target is found within the increment, it goes to that target location. The starting point is shown with a red pin, and final location is shown with a blue pin. Targets are shown with red dots.

Chapter 4

Cooperative Foraging in Cells

4.1 Introduction

Collective behavior is an emergent phenomenon consisting of large groups of autonomous individuals that interact and communicate locally, but form collective groups capable of decision making collectively [3, 78]. This phenomenon occurs in different scales from bacteria to fish and birds [3, 38, 41, 79]. This behavior extends to cells and it has been shown that clusters of cells exhibit collective motion during tissue development and repair, and tumor growth [80, 81]. These systems show long-range, scale-free correlations and discontinuous phase transitions [44]. Collective motion of these groups was found to exhibit three distinctive phases: running, rotating, and random [37, 82, 83]. In the running phase, agents are mostly aligned, and the group's center of mass moves with a large translational velocity. For the random phase, agents velocities are uncorrelated, and the group motion is minimum. In the rotating phase, group rotates around a common center. The reasons behind the emergence of the rotating phases are less clear than running and random phase. Mechanisms such as confinement and long-range interactions can result in rotating phases [45, 84].

Cell aggregation and motility are observed in various types of cancer such as lymphomas and breast cancer [49]. Lymphomas migrate in clusters and spread to distant parts of the body. Studies based on mice breast cancer, show cell clusters having higher metastatic potential [85], and it has been shown that collective cell migration is influenced by chemical signaling [86], physical cues [87] such as cell-cell adhesion

[88], and contact inhibition [89]. Therefore, it is important to understand the biophysical mechanisms that underlie the motility, sensing, and navigation of these clusters. To study this system, a model is needed such that it is capable of making connections between the complex emergent cluster behaviors, functionality and the underlying single cell properties over a large range of experimentally adjustable parameters. The motility of malignant lymphocyte clusters in chemokine gradients was studied recently [20]. Three collective phases, running, rotating, and random are also observed in clusters of malignant lymphocytes.

Malignant lymphocytes migrate individually and form clusters, and look for source of chemoattractant. In moderate gradients both individual cells and clusters move up the gradient towards higher concentrations of chemokine. However, in a steep gradient, individual cells move backwards or move towards lower concentrations, while clusters still move along the gradient. Leader cells at the front of the cluster are more motile and responsive to chemokine. In steep gradients single cells receptors get saturated and they start going towards lower concentrations of chemokine, and chemo-repulsion happens. However, it is speculated that rotations and shuffling of cells between core and rim in the random phase help the clusters in chemotaxis. The cluster rotates and then a fresh cell becomes the new leader, or in the random phase, they shuffle and new cells with fresh receptors come to the cluster edge, and the cluster can keep sensing the chemical gradient and search for the source of chemoattractant [62].

Using an agent-based model, it was shown that a possible mechanism behind the rotating phase is density-dependent cell propulsion due to contact inhibition of locomotion, whereby cell protrusions are inhibited by adhesion between cells [20]. The model only allows short-range, nearest neighbor interactions and unconfined space. The contact inhibition causes cells at the core of the cluster to move slower compare to the cells at the rim, since they have more neighbors. Therefore, cells at the rim move faster than the core cells and display stronger alignment interactions. A uniform cluster with identical cells remains in one phase throughout the simulation, either random or running, depending on the stochastic angular noise (from normal distribution) and propulsion of cells. No rotation occurs.

They showed that decoupling the rim and core suppress rotational phases, thus the coupling of these two leads to rotation. When the rim cells are in an ordered state

with respect to their velocity alignment and the core is disordered, coupling of them causes a frustrated state where the ordered rim is being pinned by the disordered core. In this state, the cluster is not able to migrate in a running phase. In order to relieve the frustration, the ordered rim starts pulling the disordered core. Resulting in the rotation of cluster around a center. The agent-based swarming model captured the dynamics of each collective phases, and the time spent in these phases successfully. It was also shown that the cluster spends most of its time in the running phase, and less in the rotating phase when a chemical gradient is introduced, which has been seen experimentally. Moreover, by increasing cluster size, the proportion of time spent in the rotational phase increases. When the cluster size increases, time spent in the rotational phase increases, and the proportion of running phase decreases which results in a decrease in cell exchanges [20].

As the gradient increases these clusters spend most of their time in the running phase which resembles a bacteria run and tumble motion. This makes the system a good candidate for cooperative foraging. However, an increased running phase means the random phase, and the rotating phase will be suppressed. The speculation was that random and rotating phases help clusters chemotax more robustly. Decreased rotations mean less load sharing which means leader cells receptors could get saturated and chemo-repulsion may occur. But, it was shown that exchanges of cells between rim and core mostly happen in the running phase, and a high rate of exchange of cells between rim and core was observed [20]. Now, the question is, how does the cluster allow cells to exchange in the running phase? In the presence of chemo-repulsion, does the running phase help clusters chemotax and find the source of chemoattractants?

With these results and knowing the mechanisms that cause the collective phases, I extend this model to study the foraging of cells for chemoattractants. Previous model showed the phases and behavior that are needed to overcome chemo-repulsion, but it did not include chemo-repulsion. By using agent-based simulations which account for chemo-repulsion, I show that by increasing the chemical gradient, clusters follow the gradient consistently, and the chemotactic efficiency increases, while individual cells and smaller clusters experience chemo-repulsion which we verify through experiments. I also find that there are three unique structures, single vortex, double vortex and disordered structures within the cluster when the cluster is in the running phase which allows the cluster to exchange cells between the rim and core and continue to

chemotax and avoid chemo-repulsion. Furthermore, I show that exchanges can occur at any location in the rim, and there is no preferred location or direction. Finally, the time cells spend in the clusters rim between the times they spend in the core is non-monotonic and it increases by increasing the gradient to moderate values, and it decreases in high gradients due to chemo-repulsion.

4.2 Simulation Model

Cell clusters are modeled as groups of particles that move with over-damped (inertia free) dynamics in two dimensional continuous space. Cells are initialized on a circular disk with velocities pointing in randomized directions. At each time step, Δt, in the simulation, each cell's position is updated according to its respective velocity, $v_i(t)$ (Eq. (4.1)). Cells velocities, (v_i), are determined by their internal self-propulsion (with magnitude p_i), as well as physical interactions between cells such as adhesions, and collisions with neighboring cells, which are modeled with a Lennard-Jones force $(\overrightarrow{LJ})$. r is the average size of the two interacting cells and $\overrightarrow{d}_{ij}$ is the separation vector between them (Eq. (4.3)).

A spring-like surface tension, long-range interaction $\overrightarrow{S}$, keeps the cells together and suppresses disaggregations. The spring interaction exists between $n.n.n.$ which are next nearest neighbors, and σ_j takes a value of 0 when there is a cell between cell i and next nearest neighbor j, and 1 otherwise (Eq. (4.4))(Fig. 4.1 a).

Cells propel themselves in a direction $(\hat{\boldsymbol{n}})$, with a self-propulsion strength p, that is determined by the memory of their own previous polarization $(\hat{\boldsymbol{v}})$, and an alignment interaction with the mean orientation of neighboring cells, $(\hat{\boldsymbol{V}})$, with interaction strength α, which captures the tendency of cells to adopt a polarity similar to their neighbors (Eq. (4.5)).

Moreover, $\hat{\boldsymbol{n}}$ depends on a gradient force (Eq. (4.2)) where the sum j is over each distinct pair of adjacent neighbors of cell i. $\overrightarrow{f_j}$ is a vector pointing in the direction bisecting the angle subtended by the centers of the cells of the neighbor pair at the center of cell i, with a magnitude equal to the arc length between the two neighbors (Fig. 4.1 b). Here, g reflects the strength of the influence on propulsion direction from the chemokine gradient per unit distance of exposed cell edge arc length, c' is the change in chemokine concentration per unit distance, and y is the distance

(in micrometers) from a concentration point of 0 ng/ml. When the magnitude of the gradient force (Eq. (4.2)) is higher than a threshold, the cell tends to go to the opposite direction due to chemo-repulsion.

Finally to capture the inherent noise within these clusters, due to the randomness of forces caused by cystoskeletal fluctuations, substrate-adhesion irregularities etc., I add a uniform and uncorrelated noise term, $\eta(t)$, to the total force each cell experiences (Eq. (4.6)). All of the interactions that the cells feel assumes cells communicate with each other by contact. So, cells interact with each other within a distance slightly larger than an average single cell diameter (r). The cell diameters are chosen from a Gaussian distribution, as normal distributions lead to cell clusters that are crystalline, and sizes have positive values.

The effect of contact inhibition of locomotion is implemented by scaling the propulsion inversely with the number of neighbors. Core cells have lower propulsion, since they form fewer protrusions (with average of 6 neighbors) compared to rim cells that have more open space, and less number of neighbors with average of 3.67 neighbors (Eq. (4.7)) [20, 90]. Parameters used in the simulations are given in (Table. 3.1).

$$\overrightarrow{x_i}(t+\Delta t) = \overrightarrow{x_i}(t) + \overrightarrow{v_i}(t)\Delta t \tag{4.1}$$

$$\overrightarrow{g}_i = gc'y \sum_j^{p.a.n} \overrightarrow{f_j}, \quad \text{if} \quad \overrightarrow{g}_i > \gamma : \overrightarrow{g}_i \rightarrow -\overrightarrow{g}_i \tag{4.2}$$

$$\overrightarrow{LJ} = -12 \sum_j^{n.n} [\frac{r^{12}}{d_{ij}^{13}} - \frac{r^6}{d_{ij}^7}] \hat{d_{ij}} \tag{4.3}$$

$$\overrightarrow{S} = \sum_j^{n.n.n} \sigma_j \overrightarrow{d}_{ij} \tag{4.4}$$

$$\hat{n} = \frac{\hat{v}(t-\Delta t) + \alpha \hat{V} + \overrightarrow{g}}{|\hat{v}(t-\Delta t) + \alpha \hat{V} + \overrightarrow{g}|}, \quad \hat{V} = \frac{\sum_{n.n} \overrightarrow{v_i}(t)}{|\sum_{n.n} \overrightarrow{v_i}(t)|} \tag{4.5}$$

$$\overrightarrow{v_i}(t) = p\hat{\boldsymbol{n}} + \epsilon \overrightarrow{LJ} + k \overrightarrow{S} + \overrightarrow{\eta} \tag{4.6}$$

$$p_i = p_{core} + \frac{3}{7}(p_{core} - p_{rim})(n_i - 6), \quad n_i : \text{ number of neighbors around the cell i} \quad (4.7)$$

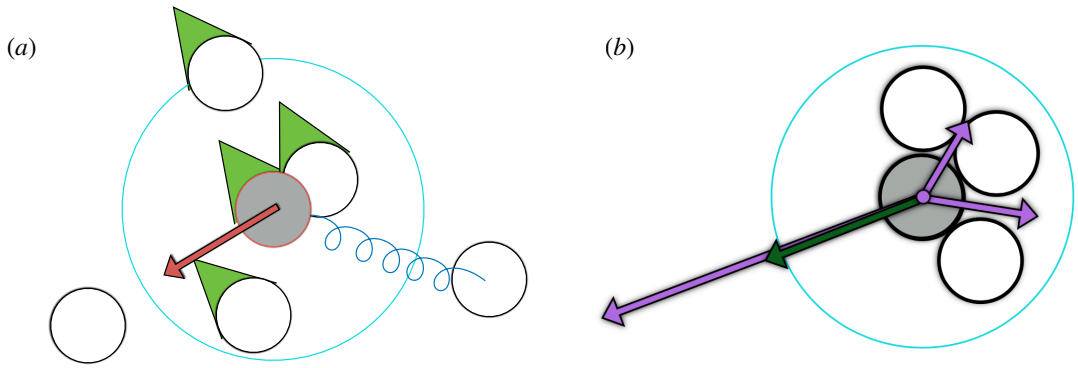

Figure 4.1: (a) Schematic for the model. Green direction indicators show the direction of the neighbors of the gray cells, and the green indicator on the gray cell shows the alignment interaction ($\overrightarrow{V}$). The red arrow is the total Lennard Jones interaction ($\overrightarrow{LJ}$) on the gray cell. Finally, the blue spring denotes the cell-cell adhesion interaction ($\overrightarrow{S}$). Note that it only exists between the gray cell and its second nearest neighbors that do not have cells interrupting the path between them. (b) Schematic illustration of the chemical gradient force on the gray cell. The purple arrows show the force between each pair of adjacent neighbors while the green arrow shows the over all outward gradient force

4.3 Results

4.3.1 Efficiency of the search for chemoattractants in clusters

It was shown that in the case of high gradients, small clusters and single cells travel to lower concentrations of Chemokine while the clusters can robustly move towards higher concentrations [62]. To determine how do the clusters follow the gradient as the

gradient increases, and how do their direction of motion changes over time in higher gradients I look at the forward migration index and velocity-velocity correlation in both experiment and simulations. Forward migration index is defined as time spent traveling up the gradient over total time. This resembles the efficiency of the search for chemoattractants.

The previous model captured different behaviors of clusters that are needed for robust chemotaxis and load sharing, such as the clusters collective phases, running, rotating and random, and the amount of time clusters spend in each of these phases. However, it did not include chemo-repulsion [20, 90]. We added chemo-repulsion to the previous model, and were able to capture the clusters behavior in high gradients.

We find that the forward migration indexI (Eq. (4.9)) increases as the gradient increases, and clusters spend most of their time migrating to higher concentrations of chemokine. However, in very high gradient forward migration index decreases slightly, and clusters go back to lower concentrations from time to time which is a result of chemo-repulsion (Fig. 4.2 a). The experimental data verifies our findings with forward migration index increasing as the gradient increases, and there is a small drop in very high gradients (Fig. 4.2 b).

Parameter	**Description**	**Value(s)**
N	Number of cells	19 to 91
Δt	Time step	0.01
α	Strength of alignment interaction	6
$\lvert\overrightarrow{\eta}\rvert$	Noise magnitude	4
ϵ	Strength of Lennard-Jones interaction	18
k	Strength of spring force	0.1
p_{core}	Propulsion value for core cells	4
p_{rim}	Propulsion value for rim cells	8

Table 4.1: Simulation parameters.

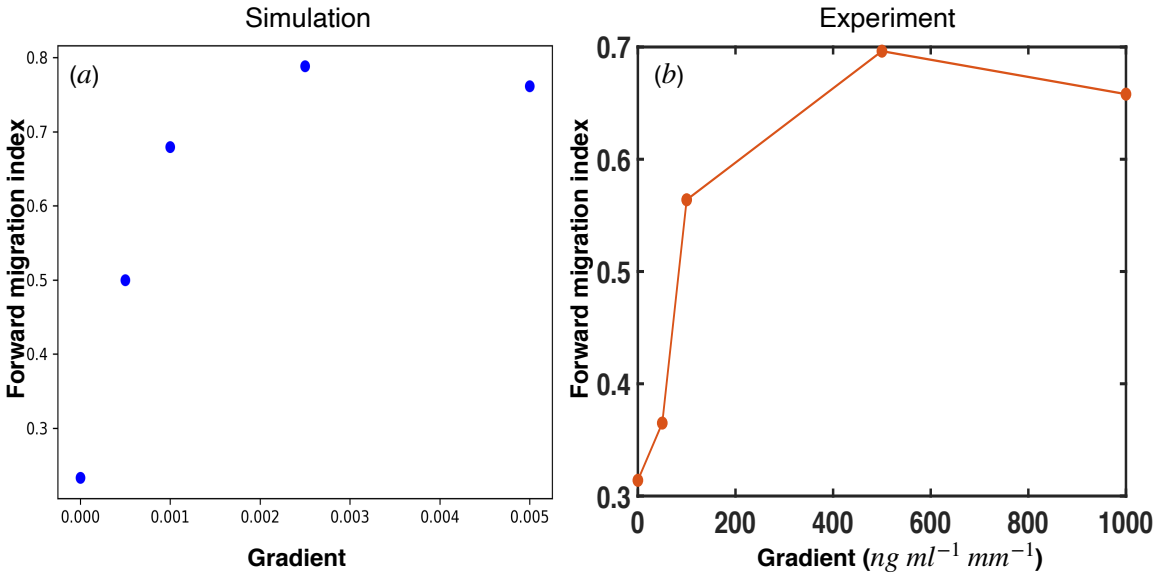

Figure 4.2: (a) Forward migration index of the clusters as a function of the chemical gradient in the simulations. The time clusters spend traveling to higher concentrations of chemokine increases as gradient increases. However, in very high gradients, it drops slightly due to chemo-repulsion affecting the clusters. (b) Forward migration index of the clusters as a function of the chemical gradient in the experiments. The same behavior is observed where the forward migration index increases with increasing the gradient and slightly drops in high gradients.

We observe that as we increase the gradient, the velocity-velocity correlation drops slower, and clusters are able to follow a consistent direction for a longer time which is in fact the direction in which the concentration increases. However, when the gradient is very high, the direction becomes uncorrelated faster compared to moderate gradients which is due to chemo-repulsion affecting the clusters (Fig. 4.3 a) which is in agreement with experiments (Fig. 4.3 b).

4.3.2 Internal dynamics of the clusters

The exchanges of cells between rim and core mostly happen in the running phase [20]. To understand the physics of load sharing inside of the clusters and the intracellular dynamics which allow the clusters to chemotax, I look at the velocity field of the clusters in the center of mass frame when the cluster is in the running phase. I observe three structures in the center of mass frame: a single vortex structure, a

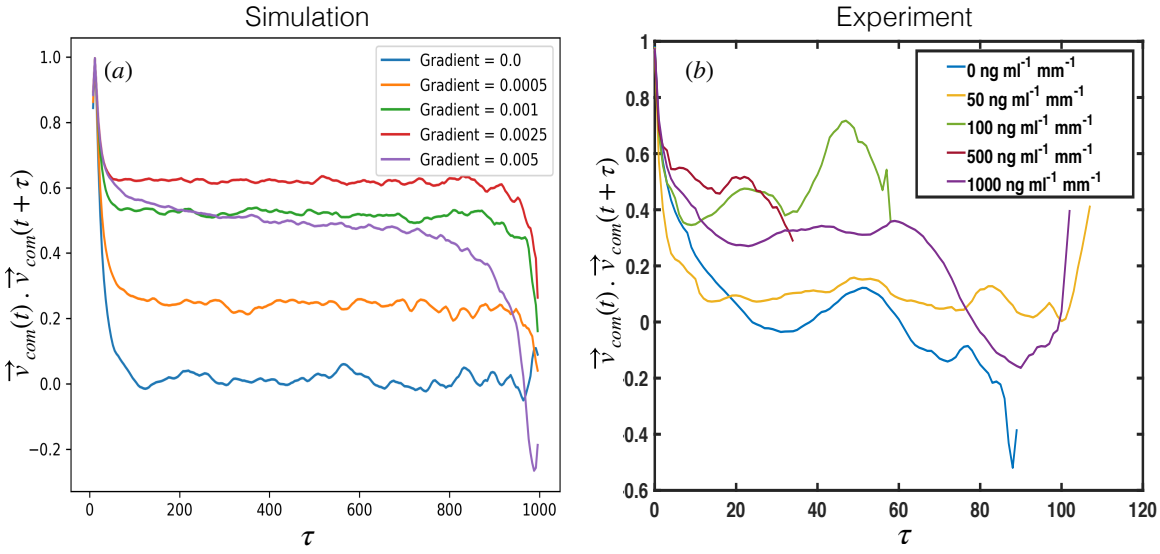

Figure 4.3: (a) Velocity-velocity correlation of the clusters center of mass in simulations for various gradients. The velocity-velocity correlation increases overall and drops slower as the gradient increases except when the gradient is high (purple curve) which is due to chemo-repulsion affecting the clusters. (b) Velocity-velocity correlation of the clusters center of mass in the experiments for various gradients. The velocity-velocity correlation has a similar behavior to the experiments and it shows the faster drop in high gradients.

double vortex structure and a disordered structure.

We see that the cluster is in the running phase in the lab frame, and cells are highly aligned together, and are going to the same direction. However, when we subtract the center of mass velocity, we observe is a single vortex, and cells rotate around the cluster's center of mass. The direction of the flow can be clockwise or counterclockwise, and it occurs in different gradient values and cluster sizes (Fig. 4.4). Looking at the experiments velocity fields, we see similar structures with vortices going clockwise and counterclockwise in the center of mass frame (Fig. 4.5).

Moreover, we see a double vortex structure. The clusters are in the running phase in the lab frame, and cells velocity vectors are highly aligned. Considering the front of the cluster as the direction of center of mass velocity, the internal dynamics of the clusters show a double vortex structure where cells in the front of the cluster move towards inside and cells in the back of the cluster move outside of the cluster (Fig.

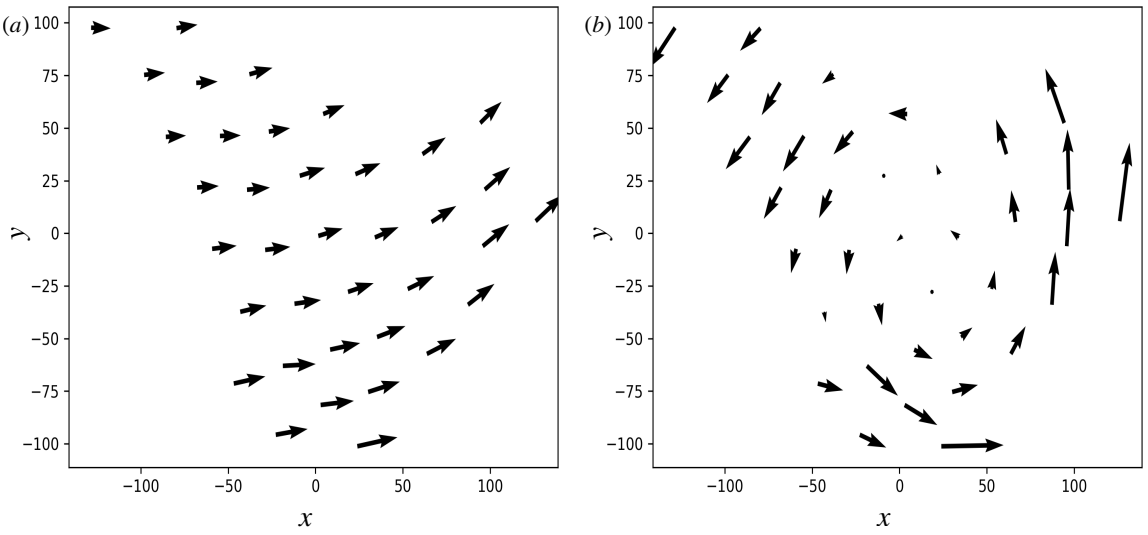

Figure 4.4: (a) Velocity field of the cells in the lab frame in the simulations. Cells are in the running phase. (b) Velocity field of the cells in the center of mass frame. Cells form a single vortex structure and rotate around the cluster's center. The direction of the rotation can be clockwise or counter clockwise. This structure allows the clusters bring fresh cells to the cluster's rim to continue to chemotax.

4.6). However, in the experiment, the cells are able to cells come out from the front of the cluster, and go inside from the back, and come out and go back from the sides (Fig. 4.7).

Finally, when the cluster is in the running phase, there is a disordered structure in the center of mass frame, and there is no distinct flow (Fig. 4.8). All three of these structures allow the cells to exchange between the rim and core. A single vortex, double vortex or a disordered structure, can bring cells with fresh receptors to to the clusters rim to avoid chemo-repulsion.

Exchanges are speculated to bring fresh cells to the cluster's leading edge. Knowing the flows inside the clusters and how they help with exchanges, we look at the exchanges at the boundary/rim of the clusters to see if there is a specific location where the exchanges happen, and to see if the cells mostly go from the rim to core or from the core to the rim based on their location in the cluster. A cell that goes from rim to core would be considered a -1 exchange, and a cell going from core to rim would be a $+1$ exchange (Fig. 4.9 a).

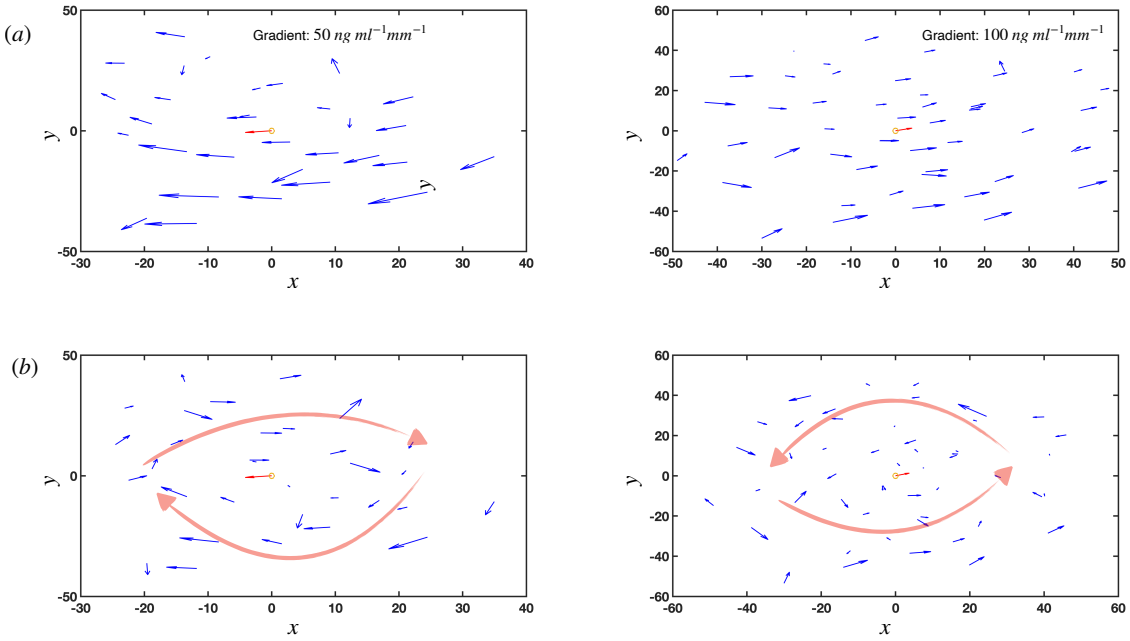

Figure 4.5: (a) Velocity field of the cells in the lab frame in the experiments. Cells are in the running phase. The units for the axis are in μm. (b) Velocity field of the cells in the center of mass frame. Cells form a single vortex structure and rotate around the cluster's center. The direction of the rotation is marked with red arrows, and it can be clockwise or counterclockwise. This structure allows the clusters bring fresh cells to the cluster's rim to continue to chemotax.

Considering the leading edge as the direction of the center of mass velocity, and dividing the cluster to four quadrants, back, front, left and right (Fig. 4.9 a), we count the number of defects in each quadrant and count the number of cells that go from rim to core, and the number of cells that go from core to rim. We find that independent of the gradient value, exchanges happen everywhere on the rim and there is no preference for their location and their sign. 50% of the time they go from rim to core and 50% they go from core to rim in agreement with the experiments (Fig. 4.9 b,c).

Finally, we look at the amount of time spent by each cell in the rim, between two core events. We observe when the gradient is zero, cells tend to stay in the rim for a shorter time and they exchange more frequently due to randomness. However, when the gradient increases to low and moderate, the amount of time spend in the rim until they go to the core increases since the cluster is more organized moving in a specific direction. Finally, when the gradient is high, the time reduces due to chemo-repulsion

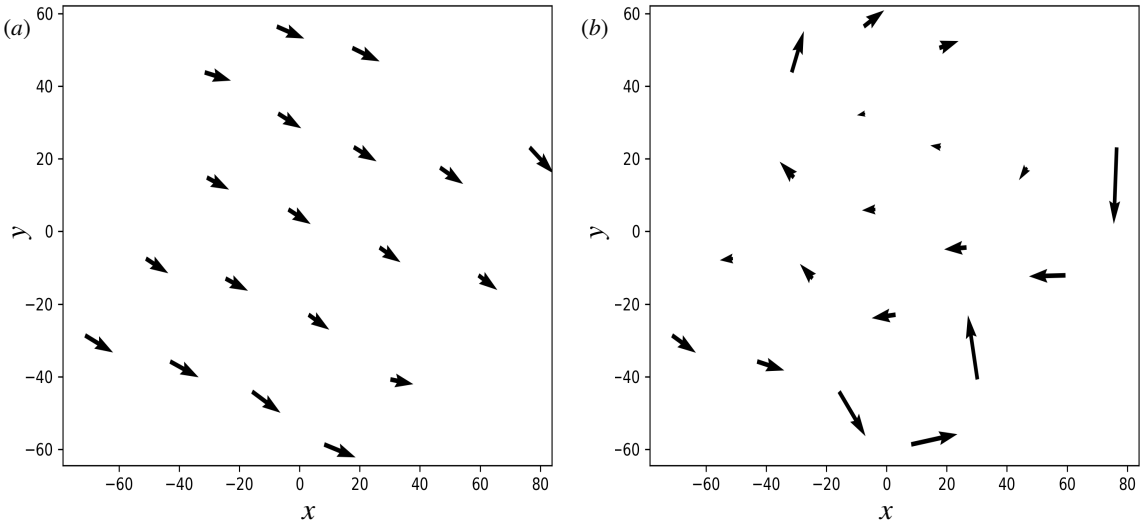

Figure 4.6: (a) Velocity field of the cells in the lab frame in the simulations. Cells are in the running phase. (b) Velocity field of the cells in the center of mass frame. Cells form a double vortex structure. The vortices only rotate in the opposite direction and cells move inwards in the front and outwards in the back. This double vortex structure can lead to exchanges between the rim and core of the cluster.

and cells going backwards to the rim (Fig. 4.10 a,b). Furthermore, the time spent in the core by individual cells decreases monotonically as gradient increases. This suggests that in lower gradients, the exchanges are mostly between the cells closer to the rim, while in higher gradients, the cells from the core and middle of the clusters will also exchange more frequently, and get a chance to be in the rim which can help the clusters to continue to forage for source of chemoattractants (Fig. 4.11 a,b).

4.4 Discussion

Collective behavior is observed across scales from cells to fish and wildebeest [3], and it allows groups to make decisions and complete tasks [3, 78]. Collective chemotaxis, foraging of cells in the direction of increasing chemical gradient, plays an important role in various biological processes such as tissue development and cancer metastasis [80, 91]. It has been shown that endothelial cells, lymphoid, and neural crest cells move collectively and search for chemoattractants [62, 92, 93].

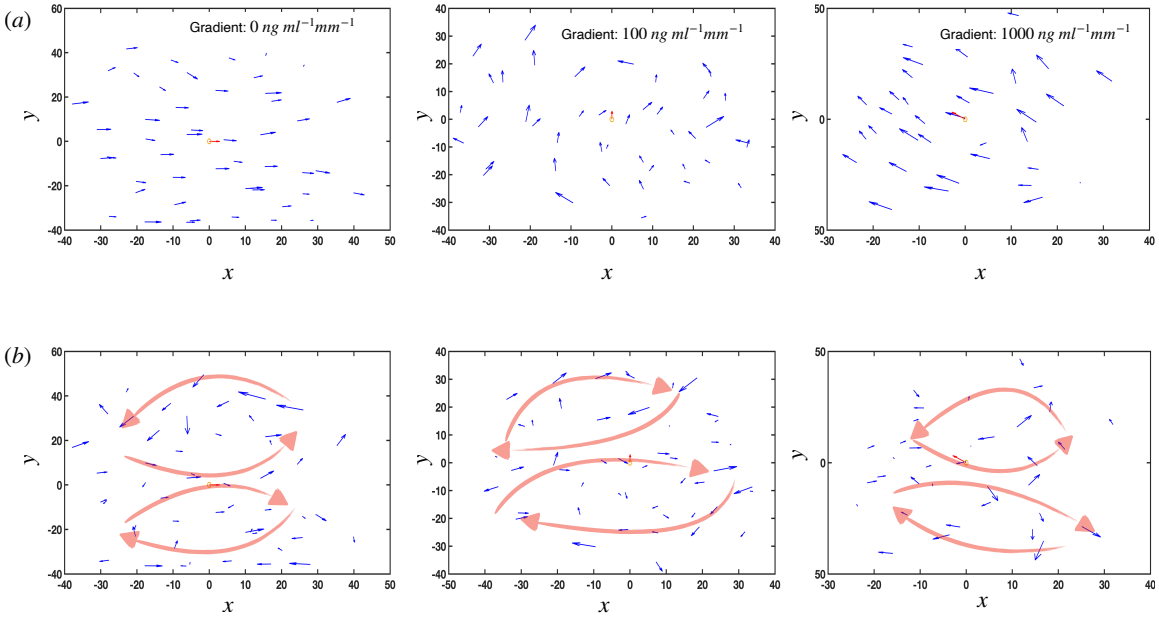

Figure 4.7: (a) Velocity field of the cells in the lab frame in the experiments. Cells are in the running phase. The units for the axis are in μm.(b) Velocity field of the cells in the center of mass frame. Cells form a double vortex structure. The vortices rotate in the opposite direction, or in the same direction. Cells can move inwards in the front and outwards in the back. They can also move inwards in the back, outwards in the front, as well as moving in and out from the sides. This double vortex structure can lead to exchanges between the rim and core of the cluster.

Malignant lymphocytes looking for source of chemokine is an ideal example of cooperative foraging. Individual cells come together and form clusters. It is shown that B & T lymphocytes are not capable of following high gradients, and tend to travel to lower concentrations of chemokine while clusters are able to move towards higher concentrations of chemokine. Clusters exhibit three distinct collective phases, running, rotating and random which are speculated to enhance the chemotaxis by bringing fresh cells to the clusters leading edge through rotation and shuffling between the periphery and the core in the random phase. An agent-based model was used to show the physics behind these phases. This model showed the phases and characteristics that are necessary to overcome chemo-repulsion [20, 90]. By adding chemo-repulsion to this model, we are able to simulate all the characteristics of the clusters (i.e. phases of motion) as well as clusters behavior in presence of high gradient. We look at forward migration index and clusters center of mass velocity-velocity

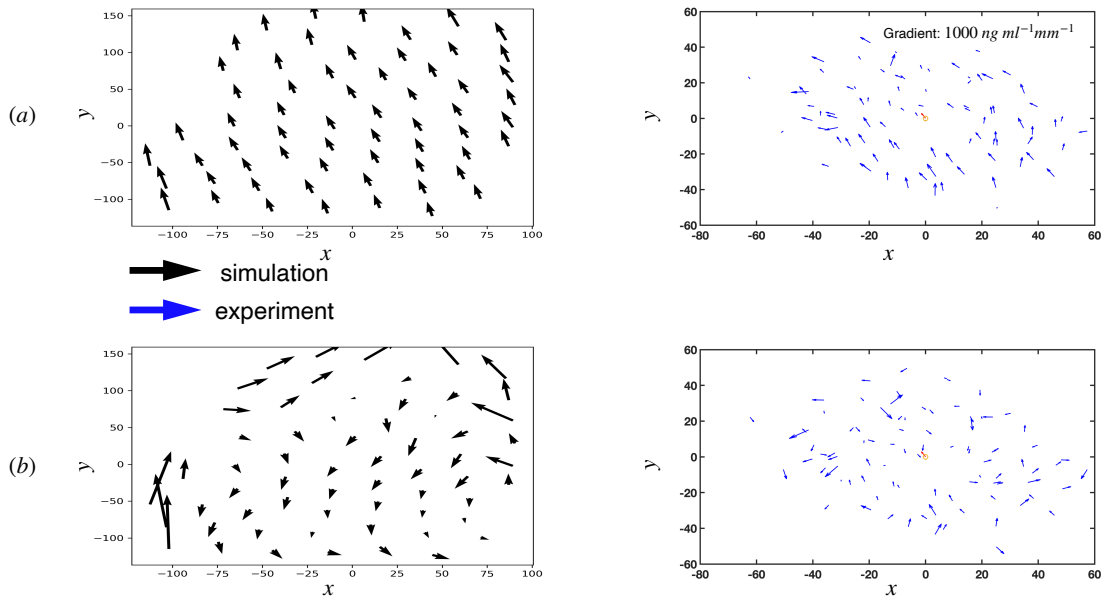

Figure 4.8: (a) Velocity field of the cells in the lab frame. Cells are in the running phase in both simulation and experiment. (b) Velocity field of the cells in the center of mass frame. Cells are in a disordered state, and no specific flow is observed. This disordered structure can shuffle the cells between the rim and core of the clusters.

correlation to see how the clusters motion changes by increasing the gradient. It turns out, aggregates of lymphocytes tend to spend longer times following the gradient as the gradient increases, and in the case of very high gradient, they will be affected by chemo-repulsion and their forward migration index slightly decreases. Moreover, the velocity-velocity correlation drops slower when the gradient is increased, and in higher gradients it slightly drops faster due to chemo-repulsion. This shows that clusters are able to move persistently towards higher concentrations of chemokine while single cells experience chemo-repulsion and travel backwards.

Furthermore, internal dynamics of the clusters in the running phase show three structures: single vortex, double vortex and disordered structures. Single vortex structures can be clockwise or counterclockwise in both experiments and simulations. In the experiments, double vortex structures can rotate in the same direction, or in two different directions, and cells can come out of the cluster from the front, back and the sides. However, in the simulations we only observe vortices rotating in the opposite direction, and cells going inside in the front and coming out from the back. This could be due to factors such as noise, or a different mechanism for chemo-repulsion in the

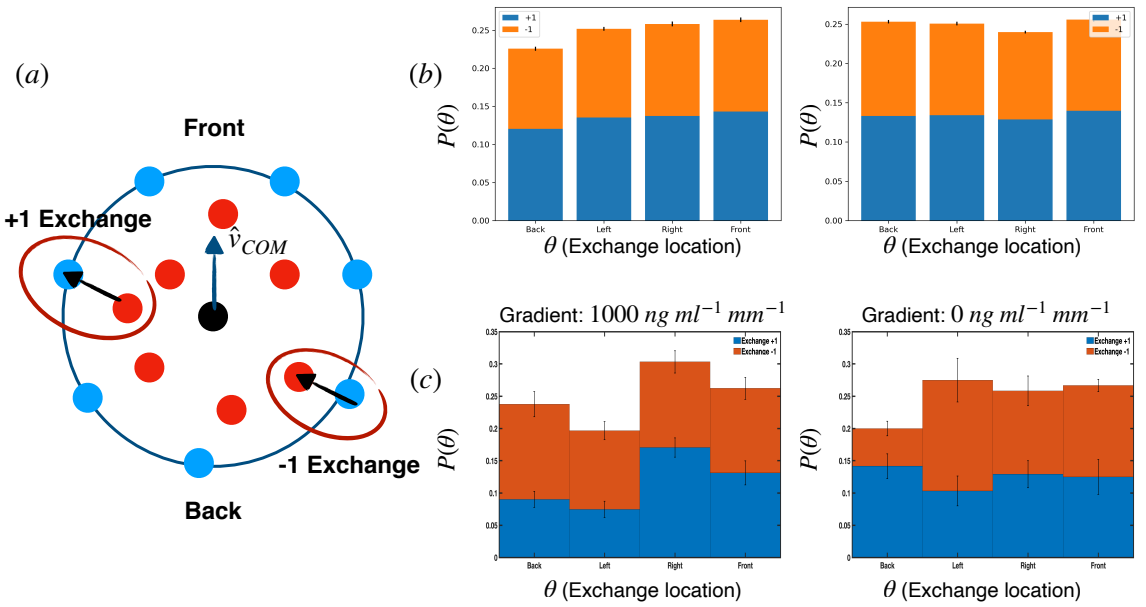

Figure 4.9: (a) Sketch of exchanges in the cluster. The blue disks are the rim cells, and the red disks are core cells, and the black disk is the center of mass. A cell going from the core to the rim is considered a $+1$ exchange. A cell going from the rim to the core is considered a -1 exchange. (b) Exchanges location with respect to the center of mass velocity in the simulations for high and low gradients. In both cases the exchanges can be found everywhere on the rim and there is no preferred location. The sign of the exchanges are $+1$ half of the times and -1 half of the times, and it does not depend on exchanges location. (c) Location of the exchanges in the experiments. The same behavior as simulation is observed. The exchanges are equally distributed in different parts of the rim and there is no preference for their sign.

experiments. Currently the model checks the gradient experienced by each cell, and if it is higher than a threshold, it flips the direction of the gradient vector. Perhaps an integrating mechanism is needed which keeps track of the gradient experienced by cells over time, and once they are saturated, it switches to chemo-repulsion. Finally, disordered structures are observed in both simulations and experiment, and they show no distinct flow within the clusters. These three structures cause cells to exchange at the rim, and allow clusters to continue to chemotax in the running phase. Pattern recognition and image analysis techniques could be applied to the clusters velocity fields to detect these structures, and measure the proportion of time clusters spend in each of the structures. Such structures can be quantified based on the clusters size

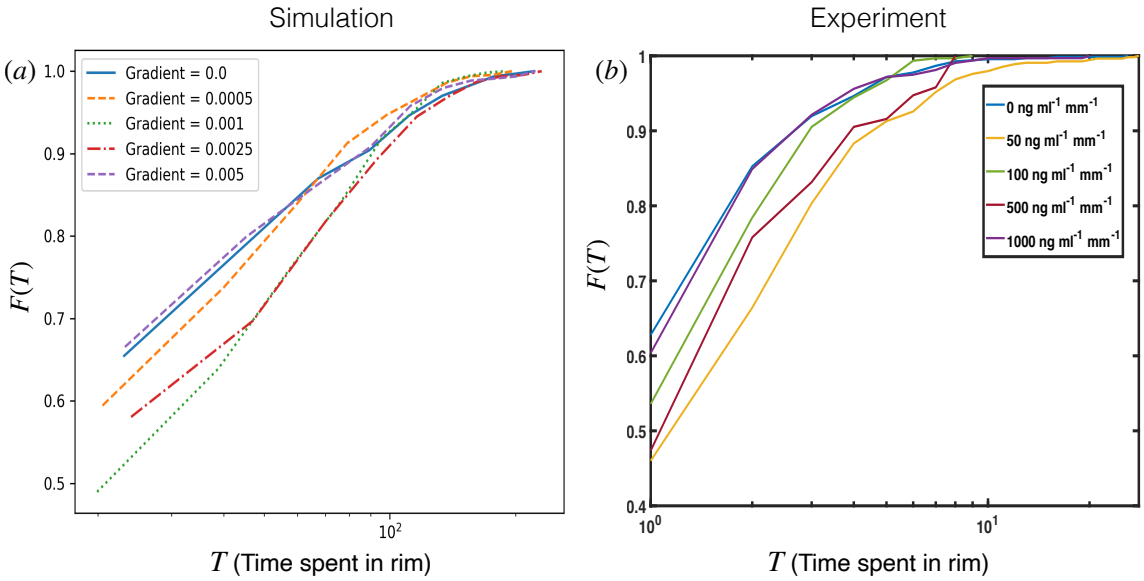

Figure 4.10: (a) Cumulative distribution function of the times cells spend in the rim between the times they were in the core in the simulations. Individual cells spend shorter times in the rim when there is no gradient due to randomness. When a low to moderate gradient is added they spend longer times in the rim since the cluster becomes more ordered. Then, in higher gradients they spend shorter times in the rim before going back to the core due to chemo-repulsion. This means more exchanges occur between the rim and the core which can help the clusters to continue to chemotax. (b) Cumulative distribution function of the time cells spend in the rim between the times they spend in the core in the experiments. A similar trend as the experiments is observed.

to see whether a specific structure is favored for a specific cluster size.

Moreover, there is no preference for the exchanges location. They occur everywhere at the rim of the cluster, and there is no specific direction for the exchanges based on their position. The probability of the cells going from rim to core, or core to rim is 50% all around the clusters rim. This suggests that the main internal mechanism behind the exchanges could be the disordered structures or double vortex structures because they have different directions of flow in the experiments. Additionally, the time individual cells spend in the clusters rim, has a non-monotonic behavior. Cells spend short times in the rim when there is no gradient perhaps due to randomness. They spend longer times in low to moderate gradient because they clusters become more organized and move to the same direction. And, they spend

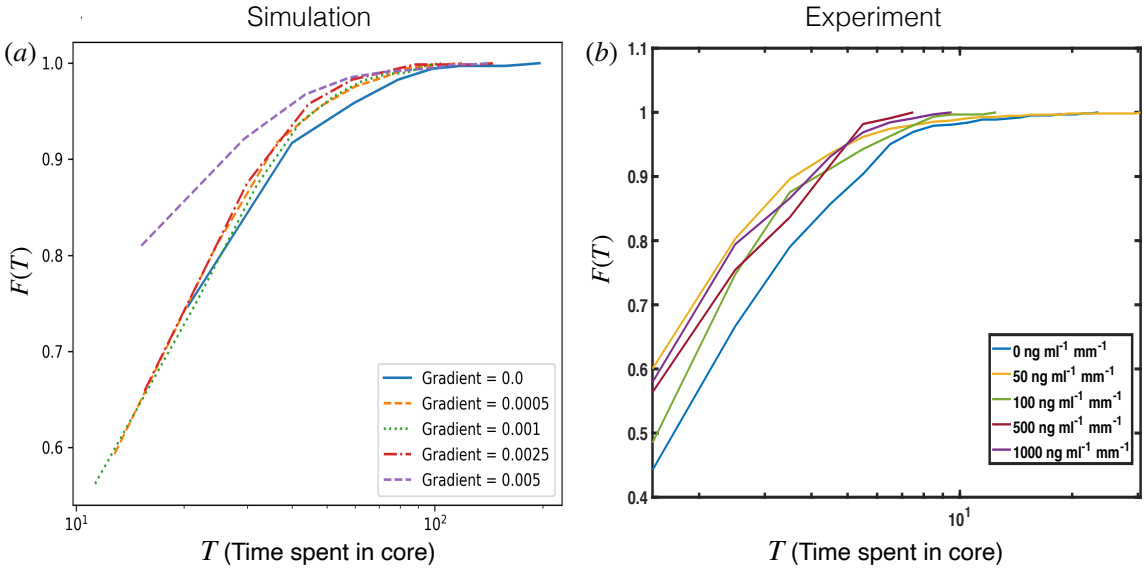

Figure 4.11: (a) Cumulative distribution function of the time cells spend in the core between the times they were in the rim in the simulations. The times get shorter as gradient increases suggesting that in lower gradients the exchanges are likely happening between the cells that are closer to the rim. However, in higher gradients, cells from the middle of the cluster will also be brought to the rim. (b) Cumulative distribution function of the time cells spend in the core between the times they spend in the rim in the experiments. A similar trend as the simulations is observed.

shorter times when the gradient is high which could be due to chemo-repulsion and cells going inside of the cluster. The time that cells spend inside the core, has a monotonic behavior, and it decreases by increasing the gradient. These results suggest that when there is no gradient, rim cells mostly switch with the cells closer to the periphery, and in higher gradients cells that are farther from the rim also will be exchanged and brought to the rim to avoid chemo-repulsion.

4.5 Appendix

4.5.1 Measuring the forward migration index

The forward migration index (FMI) as mentioned in section §4.3.1 is a measurement of time spent traveling towards the higher concentration of chemoattractant over total travel time. In the simulations and experiments, the gradient is in $+y$ direction, and the concentration increases as y increases. To measure the forward migration index, I sum over the projection of unit vector of velocity of the cluster center of mass, $\hat{v}_{COM}$ along the $+y$ axes (Eq. (4.8)), and divide it by the number of time steps in the simulations, or number of frames in the experiments (Eq. (4.9)):

$$\nu_t = \hat{v}_{COM} \cdot \hat{j} \tag{4.8}$$

$$FMI = \frac{\sum_{t=1}^{T_{total}} \nu_t}{T_{total}} \tag{4.9}$$

Since ν is equal to the *cosine* of the angle between $\hat{v}_{COM}$, and $\hat{j}$, it always fluctuates between $[-1, 1]$. If the cluster is going mostly towards higher concentrations of chemoattractant, FMI (Eq. (4.9)) will be close to 1, and if it is going towards lower concentrations, the FMI will be close to -1 or negative.

4.5.2 Distinguishing between rim and core cells and tracking them

In this part, I will cover how to distinguish between rim and core cells, and track the cells in simulations and experiments.

Distinguishing between rim and core cells

The method for finding the cells position in the cluster is similar to [20]. First, we need to find the nearest neighbors of the each cell. The nearest neighbors are cells within $\sim 1.3r$ of a cell, where r is the cell diameter. This can be done by going over all of the N cells in the cluster, and calculating the center to center distance of each cell, i, with all of the other $N-1$ cells. If the distance between cell i and cell j is smaller than $1.3r$, cell j will be added to the neighbor list for cell i. This operation has a computation time complexity of $\emptyset(N^2)$. However, since the clusters are fairly small,

the computation time does not take very long. After finding the nearest neighbors for each cell, a circle with the radius equal to the diameter of the cell, and center located on cell's center, will be drawn. This circle goes through the neighboring cells. Next, by sorting the neighbors based on their angle with the cell, and calculating the arc length of uninterrupted segments of this circle, between adjacent neighbors, the total length of the cell's exposed edge will be calculated. Rim cells, because of having less neighbors, have a larger exposed edge compared to the core cells. If this exposed edge is greater than on cell diameter, the cell would be considered a rim cell.

Tracking the individual cells over time to measure the time spent in rim or core

For keeping track of cells positions (rim or core), I label the rim cells as 1 and core cells as 0. In the simulations, I keep track of the labels by storing them in an array for each cell over time. In the experiments, there is an array for each frame, which has each cell's index in the previous frame's arrays for position. Let us name this array A_{index}. For example, at frame t, the i_{th} index in A_{index} has the value of 36. This means the i_{th} cell at frame t, was the cell 36 at frame $t-1$. The cells can be tracked over time through the experiments by using A_{index}. However, this is more challenging since some cells disappear over time, and the arrays for their indices can have different lengths. Once I know the cells positions over time in the experiment, I make an array for each of them similar to the simulations, and keep their label (1 or 0) over the frames in that array.

For calculating the time that each cell spends in the rim, one can scan the labels array, and count the time steps that cell was marked as 1, between the times that was marked as 0. For calculating the time spent in the core, the number of time steps, or frames cell was marked as 0, between the times that was marked as 1 can be counted. For instance, for cell i, I go over the array of labels. For the first n time steps, it was in the rim and was labeled as 1. At $n+1$, it goes to the core, and now it is labeled as 0. I make a counter which starts to count the time spend in the core once the label goes from rim to core, at $n+1$. Let us say it goes back to the rim m time steps later and the label goes from 0 to 1, the value for counter, will be m, and will be stored in an array. Once the time spent in the core between two rim events is saved in the array, the counter is set to zero again, and this process continues until the end of the

array. These will produce a distribution of times spend in the rim or the core. We look at the cumulative distribution function of the amount of time spend in the rim or the core in section §4.3.2, (Fig. 4.10) and (Fig. 4.11) to see how does the time spend in the rim or core changes based on the gradient value because the probability density function depends on the binning of the data.

4.5.3 Finding the exchanges and their location

To find the exchanges, one can go through the array containing the cells labels explained in section §4.5.2. If the label changes from 0 to 1, or 1 to 0, an exchange happens. The location of this exchange, $\theta_{exchange}$, is set by the angular location of it with respect to the center of mass when the cell is on the rim. Therefore, if the cell went from rim to core, the exchange's location is the cell's initial angular position (Fig. 4.13 a), and if the cell went from core to rim, the exchange's location is the cell's final angular position (Fig. 4.13 b). Front of the cluster is the direction of the center of mass velocity, and the cluster is divided to four quadrants, front, back, left, and right. Next, by calculating the dot product and cross product of the exchanges angular position, $(\cos(\theta_{exchange}), \sin(\theta_{exchange}))$, and the unit vector of the velocity of center of mass, we can find the quadrant that the exchange is in.

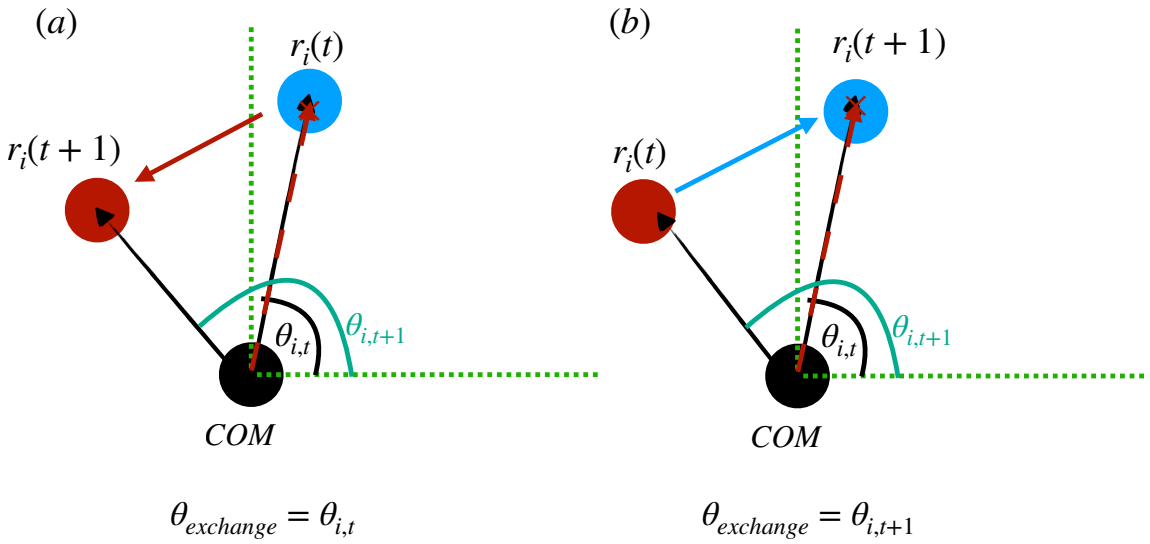

Figure 4.12: (a) Angular position of the exchange, $\theta_{exchange}$, if cell i goes from rim to core, is the cells angle with center of mass, when it was on the rim which is $\theta_{i,t}$. (b) Angular position of the exchange, $\theta_{exchange}$, if cell i goes from core to rim, is the cells angle with center of mass when it was in the rim which in this case is $\theta_{i,t+1}$. Blue disks show rim cells and red disks show core cells in both (a) and (b).

Chapter 5

Final Discussion

The objective of this dissertation was to understand how competition and cooperation affect search strategies and foraging efficiency compared to searches that are done individually. To study foraging in the presence of competition I investigated the efficient foraging strategies for territorial animals through a $2\mathbb{D}$ model and Python simulation. For cooperative foraging, I analyzed the behavior of cellular clusters such as white blood cells looking for source of chemicals.

5.1 Conclusion and future work for Chapter §3

I developed an agent-based model to study territorial competition in terrestrial and aerial animals. The searchers look for targets using Lévy flight searching patterns [2], and territories are defined as circular boundaries around each agent. Our results show that the strategies that maximize the search efficiency in terrestrial animals are similar to solitary foragers which is a combination of exploration and exploitation, however, the mean efficiency of the group decreases as the territory size or the population of foragers increases. Furthermore, aerial animals are allowed to travel longer distances and cross territories. However, resources are not available to them which results in a suppressed mean efficiency especially in smaller Lévy exponents where the probability of long jumps is higher. In addition, the deviations in the efficiency, is higher in more localized searches, which means that foraging with such strategies have a higher risk of performing below the average efficiency of the group, or a chance of performing above the average. While searching with less localized searches, would lead to the

same average efficiency with a smaller risk of performing below the average, since the probabilities of taking a longer jump and visiting new target sites are higher. Finally, we show that in the case of non-regenerative targets, the efficiency suppression in the aerial case lead to a tunable optimum in the search efficiency for larger territories, while there is no optimum observed for single searchers and terrestrial animals and efficiency decreases by increasing the Lévy exponent.

Our current results can be used to understand the natural behavior of territorial animals in nature, and can be expanded to $3\mathbb{D}$ to study aquatic animals such as sharks. Our model can also be used to study social interactions where individuals are completing a task, but they are trying to avoid interference with each other. Rescuer drones and robots usually avoid collision to minimize the over all search time[94]. Our results can also be used for such systems that are designed to explore an area, and foragers are prohibited to search the spots that others are searching.

This work can be extended to have various type of distribution for targets such as patchy or normally distributed [66, 95], or motile and stationary targets. It is known that for a single searcher, when considering a super dense environment, less super-diffusive strategies, $(2 < \mu < 3)$, perform nearly close to $\mu = 2$ [66], and in the case of sparse targets, even when targets are distributed in fragments or with a Lévy dust distribution, the most beneficial search strategy observed is Lévy flights with $\mu \approx 2$ [66]. We can implement such target distributions for territorial competitors. and compare our simulations to the results for individual foragers. Furthermore, we can implement adaptive foraging strategies, and allow the searchers to tune their Lévy exponent, or size of their territory when the search efficiency is low, or when a target patch is depleted [96].

Currently, all the foragers are identical in the model. We can also consider a more realistic system where the individuals in the group are different from each other. This can include different foraging strategies, Lévy exponents, or different territory sizes. Then, we can see how the overall efficiency changes, and if there is a strategy for individuals that can help them perform above the average of the group. Finally, we can have a combination of cooperative and competitive searchers, and allow them to switch between these interactions depending on the availability of resources and their efficiencies.

5.2 Conclusion and future work for Chapter §4

In the context of cooperative foraging, I expanded the previous work in Gopinathan's group [20] by adding chemo-repulsion to the model as well as extensively analyzing the experimental data. The previous model showed the phases and characteristics that are necessary to overcome chemo-repulsion. However, it did not account for chemo-repulsion.

We show that the in the presence of chemical gradients, clusters chemotactic efficiency increases by increasing the gradient. However, the clusters are affected by chemo-repulsion in high gradients and the efficiency slightly drops. The velocity-velocity correlation drops slower when the gradient is increased, and in higher gradients it reduces slightly due to chemo-repulsion. Moreover, we see that the internal dynamics of the clusters in the running phase reveal three different structures including single vortex, double vortex and disordered structures which allow the cells to shuffle between the rim and the core of the clusters. The exchanges occur all around the clusters periphery without any specific direction which suggests that the disordered and the double vortex are the dominant structures causing the exchanges. This shuffling brings cells with fresh receptors to the clusters rim and results in clusters robust search for chemoattractants. Finally, the time individual cells spend in the clusters rim, has a non-monotonic behavior as a function of the chemical gradient. Cells spend short times in the rim when there is no gradient due to randomness, longer times in low to moderate gradient and shorter times when the gradient is high due to chemo-repulsion.

The internal structures can be quantified based on the cluster sizes, and individual cell properties such as noise and contact inhibition of locomotion. For instance, there might be a region in parameter space which favors the double vortex structure while another region favors the single vortex structure. Methods such as measuring the vorticity and angular momentum of the system can be implemented to identify the structures. Once can calculate the circulation around the rim in the center of mass frame, vorticity, overall angular momentum, and angular momentum in front half, right half, left half and back half of the cluster. Given these quantities, the structure can be identified. For example, a single vortex has high over all angular momentum and circulation, while a double vortex will have a high angular momentum in separate two halves of the cluster. Having a method to identify the phases, proportion of

times that clusters spend in each of these structures can be measured, similar to the time they spend in each of the collective phases [20]. Furthermore, machine learning techniques could be applied to reduce the noise and enhance the velocity fields in the experimental data. In the simulation. the cells appear to be more noisy. A scan over wide variety of values for noise can be done to find a region where cells behave more similarly to the experiments.

Metastatic cancer cells such melanoma have shown Lévy walk patterns that are different than their non-invasive counter parts which perform simple diffusion. This allows the cancerous cells to migrate to distant parts of the body more efficiently[97]. A similar investigation can be done on malignant and benign lymphocytes. To spot the differences in movement patterns and trajectories, the velocity, turning angle, persistence length, and persistence time can be studied, and our model can be extended to reflect the respective behavior depending on the cell type.

An important driver of metastatic potential in the cancer cells is the propensity to form aggregates, and the stability of these aggregates in the presence of gradient. Our current model focuses on the internal dynamics and chemotactic efficiency of single clusters by suppressing any disaggregation instability with a very weak surface tension. The model can be extended such that cells will not have this constraint anymore to find the single cell properties that favor or suppress aggregation. A combined assessment of the cluster lifetimes, size distributions, shapes, and their chemotactic efficiency will lead to identifying regions of parameter space that would be excellent candidates for suppressing metastatic potential.

Appendix A

Appendix: Computer Programs Used

A.1 Introduction

In this work, the first projects simulation, numerical calculations and analysis were executed in Python; whereas in the second project the simulations were executed using C++ and all resulting data was analyzed using Python. The experimental data was analyzed using MATLAB. Here, I show the programs used to conduct the second project. The full code of simulation and analysis can be found on github, click (**Farnaz_Golnaraghi_Dissertation_Code**).

A.2 Cooperative Foraging in Cells Programs

A.2.1 align.cpp

This program calculates the alignment interactions with nearest neighbors.

```
#include <iostream>
#include <math.h>

void align(int i, int j, float angold[], float *Ax, float *Ay)
   {
 if (i!=j){
  *Ax+=cosf(angold[j]);
```

```
  *Ay+=sinf(angold[j]);
 }
 return;
}
```

A.2.2 avgvels.cpp

This program the average velocity of nearest neighbors.

```
#include <iostream>

void avgvs(int N, float x[], float y[], float xold[], float
   yold[], float *COMv, float *vavg){
 float COMvx=0,COMvy=0;
 *vavg=0;
 for (int i=0; i<N; i++){
  if (xold[i]-x[i]>L/2)
   xold[i]-=L;
  if (x[i]-xold[i]>L/2)
   xold[i]+=L;
  if (yold[i]-y[i]>L/2)
   yold[i]-=L;
  if (y[i]-yold[i]>L/2)
   yold[i]+=L;
  COMvx+=(xold[i]-x[i]);
  COMvy+=(yold[i]-y[i]);
  (*vavg)+=sqrtf((x[i]-xold[i])*(x[i]-xold[i])+(y[i]-yold[i])
     *(y[i]-yold[i]));
 }
 (*vavg)=(*vavg)/(float)N;
 (*COMv)= sqrtf(COMvx*COMvx+COMvy*COMvy)/(float)N;
 return;
}
```

A.2.3 bigarc.cpp

This program biggest arc length/exposed edge for cells.

```
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <fstream>
#define _USE_MATH_DEFINES
using namespace std;
void gradient(int i, int neigh1[], int neigh2[], int neighnum1
   , int neighnum2, float x[], float y[], float *max_diff){
 int *neigh;
 int neighnum=neighnum1+neighnum2;
 neigh=(int*)malloc((neighnum)*sizeof(int));
 for (int j=0; j<neighnum1; j++)
  neigh[j]=neigh1[j];
 for (int j=neighnum1; j<neighnum; j++)
  neigh[j]=neigh2[j-neighnum1];
 float d,dx,dy,d2,dx2,dy2;
 float *angles;
 int *jincr;
 angles=(float*)malloc((neighnum)*sizeof(float));
 jincr=(int*)malloc((neighnum)*sizeof(int));
 for (int j=0;j<neighnum;j++)
  angles[j]=10;
 for (int j=0;j<neighnum;j++){
  if (neigh[j]!=i){
   d=dist(i,neigh[j],x,y,&dx,&dy);
   float temp=atan2f(dy,dx);
   for (int k=0;k<neighnum;k++){
    if (temp<angles[k]){
     for (int l=neighnum-1;l>k;l--){
      angles[l]=angles[l-1];
      jincr[l]=jincr[l-1];
     }
     jincr[k]=neigh[j];
```

```
      angles[k]=temp;
      break;
     }
    }
   }
  }
 angles[neighnum-1]=angles[0]+2*M_PI;
 jincr[neighnum-1]=jincr[0];
 for (int j=0;j<neighnum-1;j++){
  if (jincr[j]!=i){
   if (angles[j+1] - angles[j] > max_diff):
    *max_diff = angles[j+1] - angles[j]
  }
 }
 free(angles);
 free(jincr);
 free(neigh);
 return;
}
```

A.2.4 clustering.cpp

This program using a recursive method, finds all the cells that are in the same cluster.

```
#include <iostream>

void addNewNeighbor(int i, int label[], int currentLabel, int*
    neigh[], int neighnum[]){
 int j;
 if (label[i]==-1){
  label[i]=currentLabel;
  for (j=0; j<neighnum[i]; j++){
   if (neigh[i][j]!=i)
    addNewNeighbor(neigh[i][j], label, currentLabel, neigh,
       neighnum);
  }
```

```
 }
 return;
}

void clustering(int label[], int* neigh[], int neighnum[], int
   * clusters){
 int j, currentLabel=0;
 for (j=0; j<Num; j++)
  label[j]=-1;

 for (j=0; j<Num; j++){
  if (label[j]==-1){
   addNewNeighbor(j, label, currentLabel, neigh, neighnum);
   currentLabel++;
  }
 }
 *clusters=currentLabel;
 return;
}
```

A.2.5 COM.cpp

This program finds clusters center of mass.

```
#include <iostream>
#include <math.h>

void com(int iini, int ifinal, float x[], float y[], float *
   COMx, float *COMy){
 int i;
 float avgxix=0, avgxiy=0, avgzetax=0, avgzetay=0;
 for (i=iini;i<ifinal+1;i++){
  avgxix+=cos(x[i]/L*2*M_PI);
  avgxiy+=cos(y[i]/L*2*M_PI);
  avgzetax+=sin(x[i]/L*2*M_PI);
  avgzetay+=sin(y[i]/L*2*M_PI);
```

```
 }
 avgxix=avgxix/(float)(ifinal-iini+1);
 avgxiy=avgxiy/(float)(ifinal-iini+1);
 avgzetax=avgzetax/(float)(ifinal-iini+1);
 avgzetay=avgzetay/(float)(ifinal-iini+1);
 *COMx=L*(atan2f(-avgzetax,-avgxix)+M_PI)/(2*M_PI);
 *COMy=L*(atan2f(-avgzetay,-avgxiy)+M_PI)/(2*M_PI);
 return;
}
```

A.2.6 confine.cpp

This program finds the ring confinement position.

```
#include <iostream>
#include <math.h>

void conf(int i, float x[], float y[], float COMx, float COMy,
    float *Confx, float *Confy){
 float COMxtemp=COMx, COMytemp=COMy;
 if ((x[i]-COMx)>L/2)
  COMxtemp+=L;
 if ((x[i]-COMx)<-L/2)
  COMxtemp-=L;
 if ((y[i]-COMy)>L/2)
                COMytemp+=L;
        if ((y[i]-COMy)<-L/2)
                COMytemp-=L;
 float rad=sqrtf((x[i]-COMxtemp)*(x[i]-COMxtemp)+(y[i]-
    COMytemp)*(y[i]-COMytemp));
 if (rad==0)
  rad=0.00001;

 *Confx=(rad-confring*r*3/M_PI)*(x[i]-COMxtemp)/rad;
 *Confy=(rad-confring*r*3/M_PI)*(y[i]-COMytemp)/rad;
 return;
```

```
}
```

A.2.7 distance2.cpp

This program calculates the distance between two points.

```
#include <iostream>
#include <math.h>

float dist(int i, int j, float x[], float y[], float *dx,
   float *dy){

 *dx=distx(x[i],x[j]);
 *dy=distx(y[i],y[j]);
 return sqrtf((*dx)*(*dx)+(*dy)*(*dy));
}
```

A.2.8 distx.cpp

This program modulates x positions.

```
#include <iostream>
#include <cmath>

float distx(float x1, float x2){
// return (float)L/(2*M_PI)*asinf(cosf(2*M_PI/(float)L*x1)*
   sinf(2*M_PI/(float)L*x2)-sinf(2*M_PI/(float)L*x1)*cosf(2*
   M_PI/(float)L*x2))/fabsf(asinf(cosf(2*M_PI/(float)L*x1)*
   sinf(2*M_PI/(float)L*x2)-sinf(2*M_PI/(float)L*x1)*cosf(2*
   M_PI/(float)L*x2)))*acosf(cosf(2*M_PI/(float)L*x1)*cosf(2*
   M_PI/(float)L*x2)+sinf(2*M_PI/(float)L*x1)*sinf(2*M_PI/(
   float)L*x2));
// return abs(asinf(cosf(2*M_PI/(float)L*x1)*sinf(2*M_PI/(
   float)L*x2)-sinf(2*M_PI/(float)L*x1)*cosf(2*M_PI/(float)L*
   x2)));//*acosf(cosf(2*M_PI/(float)L*x1)*cosf(2*M_PI/(float)
   L*x2)+sinf(2*M_PI/(float)L*x1)*sinf(2*M_PI/(float)L*x2));
        float dx=x2-x1;
```

```
        if (dx>L/2)
                dx-=L;
        if (dx<-L/2)
                dx+=L;

        return dx;
}
```

A.2.9 fluidity.cpp

This program finds the exchange rate between the rim and core.

```
#include <iostream>

void fluidity(int N, int *neigh[], int neighnum[], int *
   uniqueneigh[], int unneighnum[], int *oldneigh[], int
   oldneighnum[], int *changes){
 int *new_neigh, *temp;
 *changes=0;
 for (int i=0;i<N;i++){
  int check3=1;
  temp=uniqueneigh[i];
  for (int j=0;j<neighnum[i];j++){
   int check=1;
   for (int k=0;k<unneighnum[i];k++){
    if (neigh[i][j]==uniqueneigh[i][k]||i==j)
     check=0;
   }
   if (check==1){
    unneighnum[i]++;
    new_neigh=(int*)realloc(temp,unneighnum[i]*sizeof(int));

    if (new_neigh==NULL){
     free(temp);
     break;
    }
```

```
    else {
     temp=new_neigh;
     temp[unneighnum[i]-1]=j;
    }
   }
//////////////////////////////////////////
   int check2=1;
   for (int k=0;k<oldneighnum[i];k++){
    if (oldneigh[i][k]==neigh[i][j]&&neighnum[i]==oldneighnum[
       i]){
     check2=0;
    }
   }
   if (check2==1)
    check3=0;
  }

  if (check3==0)
   *changes++;
//////////////////////////////////////////
  uniqueneigh[i]=temp;


 }
 return;
}
```

A.2.10 gradient.cpp

This program finds the exposed edge and calculates the gradient vector (sum of arc lengths).

```
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <fstream>
```

```
#define _USE_MATH_DEFINES
using namespace std;
void gradient(int i, int neigh1[], int neigh2[], int neighnum1
   , int neighnum2, float x[], float y[], float *gradx, float
   *grady){
 int *neigh;
 int neighnum=neighnum1+neighnum2;
 neigh=(int*)malloc((neighnum)*sizeof(int));
 for (int j=0; j<neighnum1; j++)
  neigh[j]=neigh1[j];
 for (int j=neighnum1; j<neighnum; j++)
  neigh[j]=neigh2[j-neighnum1];
 float d,dx,dy,d2,dx2,dy2;
 float *angles;
 int *jincr;
 angles=(float*)malloc((neighnum)*sizeof(float));
 jincr=(int*)malloc((neighnum)*sizeof(int));
 for (int j=0;j<neighnum;j++)
  angles[j]=10;
 for (int j=0;j<neighnum;j++){
  if (neigh[j]!=i){
   d=dist(i,neigh[j],x,y,&dx,&dy);
   float temp=atan2f(dy,dx);
   for (int k=0;k<neighnum;k++){
    if (temp<angles[k]){
     for (int l=neighnum-1;l>k;l--){
      angles[l]=angles[l-1];
      jincr[l]=jincr[l-1];
     }
     jincr[k]=neigh[j];
     angles[k]=temp;
     break;
    }
   }
  }
 }
```

```
 angles[neighnum-1]=angles[0]+2*M_PI;
 jincr[neighnum-1]=jincr[0];
 for (int j=0;j<neighnum-1;j++){
  if (jincr[j]!=i){
   *gradx+=cosf((angles[j]+angles[j+1])/(float)2)*(angles[j
      +1]-angles[j]);
   *grady+=sinf((angles[j]+angles[j+1])/(float)2)*(angles[j
      +1]-angles[j]);
  }
 }
 free(angles);
 free(jincr);
 free(neigh);
 return;
}
```

A.2.11 initialize.cpp

This program initializes cells on a disk at the beginning of the simulations.

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES

void inipos(int N, int i, float x[], float y[]){
 x[i]=sqrtf((float)N)*(float)rand()/(float)RAND_MAX-sqrtf((
    float)N)+50;
 y[i]=sqrtf((float)N)*(float)rand()/(float)RAND_MAX-sqrtf((
    float)N)+50;
 float r=sqrtf((x[i]-50)*(x[i]-50)+(y[i]-50)*(y[i]-50));
 if (r>sqrtf((float)N)/10)
  inipos(N,i,x,y);
 return;
}
```

```
void initialize(int N, float x[], float y[], float vx[], float
    vy[]){
 int i;
 for (i=0; i<N; i++){
  inipos(N,i,x,y);
  vx[i]=0;
  vy[i]=0;
 }
 return;
}
```

A.2.12 initializecircle.cpp

This program initializes cells on a disk at the beginning of the simulations.

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES

using namespace std;
void initialize(int N, float x[], float y[], float vx[], float
    vy[], float xold[], float yold[], float ri[], int
   rimorcore[], int rimorcoreOld[]){
 int i=N, j=1, left=N, k, d, rings=0;

 for (i=0; i<N; i++){
  if (left>j*6){
   x[i]=L/2+6*j*r/(2*M_PI)*cosf(2*M_PI/(6.*j)*(i-1-6*(j-1)))
      ;//j*(r*N)/(2*M_PI)*cosf((float)i/(float)N*2*M_PI);
   y[i]=L/2+6*j*r/(2*M_PI)*sinf(2*M_PI/(6.*j)*(i-1-6*(j-1)))
      ;//j*(r*N)/(2*M_PI)*sinf((float)i/(float)N*2*M_PI);
  }
  else{
```

```
   x[i]=L/2+6*j*r/(2*M_PI)*cosf(2*M_PI/((float)left-1)*(i
      -1-6*(j-1)));//j*(r*N)/(2*M_PI)*cosf((float)i/(float)N
      *2*M_PI);
   y[i]=L/2+6*j*r/(2*M_PI)*sinf(2*M_PI/((float)left-1)*(i
      -1-6*(j-1)));//j*(r*N)/(2*M_PI)*sinf((float)i/(float)N
      *2*M_PI);
  }
  int n=0;
  for (k=0;k<j;k++)
   n+=k*6;
  if ((i-n)%(j*6)==0&&i>0){
   j++;
   left=N-i;
  }

  x[0]=L/2;
  y[0]=L/2;
  xold[i]=x[i];
  yold[i]=y[i];
  float initheta=2*M_PI*(float)rand()/(float)RAND_MAX;
  float iniv=vini*(float)rand()/(float)RAND_MAX;
  vx[i]=iniv*cosf(initheta);
  vy[i]=iniv*sinf(initheta);
  float u=(float)rand()/(float)RAND_MAX;
  float v=(float)rand()/(float)RAND_MAX;
  ri[i]=r+rs*sqrtf(-2*log(u))*cosf(2*M_PI*v);
  rimorcore[i]=0;
  rimorcoreOld[i] = 0;
//  if (ri[i]>R1)
//   ri[i]=0.99*R1;
 }
 return;
}
```

A.2.13 interactions.cpp

This program calculates all the interactions between cells (Lennard jones, springs, alignment)

```
#include <iostream>
#include <math.h>
#include <stdlib.h>

#include "align.cpp"
#include "LJ.cpp"
#include "spring.cpp"
#include "confine.cpp"
#include "intermags.cpp"

void interactions(int N, float grad, float noff, float pcore,
   float x[], float y[], float COMx, float COMy, float vx[],
   float vy[], int* neigh1[], int* neigh2[], int neighnum1[],
   int neighnum2[], int coupled, float ri[]){
 int i, j;
 float *angold, noise0;
 angold=(float*)malloc(N*sizeof(float));
 for (i=0; i<N; i++){
  angold[i]=atan2f(vy[i],vx[i]); //Store all the angles of all
      the dudes for alignment interaction.
 }
 for (i=0; i<N; i++){
  float Ax=0, Ay=0, A, LJx=0, LJy=0, Springx=0, Springy=0,
     Confx=0, Confy=0, gradx=0, grady=0, G, dx, dy, d, dx2,
     dy2, d2;
  for (j=0; j<neighnum1[i]; j++){
   if (((ringn(i,x,y,COMx,COMy)==confring&&ringn(i,x,y,COMx,
      COMy)==ringn(neigh1[i][j],x,y,COMx,COMy))||(ringn(i,x,y,
      COMx,COMy)<confring&&ringn(neigh1[i][j],x,y,COMx,COMy)<
      confring))||coupled==1){
```

```
    d=dist(i,neigh1[i][j],x,y,&dx,&dy);  //Calculate distance
       between i and j cells.
    align(i,neigh1[i][j],angold,&Ax,&Ay);  //Alignment
       interaction.
    LJ(i,neigh1[i][j],dx,dy,d,ri,&LJx,&LJy);  //Lennard-Jones
       interaction.
   }
  }

  for (j=0; j<neighnum2[i]; j++){
   if (((ringn(i,x,y,COMx,COMy)==confring&&ringn(i,x,y,COMx,
      COMy)==ringn(neigh2[i][j],x,y,COMx,COMy))||(ringn(i,x,y,
      COMx,COMy)<confring&&ringn(neigh2[i][j],x,y,COMx,COMy)<
      confring))||coupled==1){
    d=dist(i,neigh2[i][j],x,y,&dx,&dy);
    spr(i,neigh2[i][j],x,y,neighnum1[i],neigh1[i],&Springx,&
       Springy); //Long range spring interaction.
   }
  }
  conf(i,x,y,COMx,COMy,&Confx,&Confy); //Confinement vector.
  gradient(i,neigh1[i],neigh2[i],neighnum1[i],neighnum2[i],x,y
     ,&gradx,&grady);

  //Normalize alignment.
  A=sqrtf(Ax*Ax+Ay*Ay);
  if (A>0){
   Ax=Ax/A;
   Ay=Ay/A;
  }

//  G=sqrtf(gradx*gradx+grady*grady);
//  if (G>0){
//   gradx=gradx/G;
//   grady=grady/G;
//  }
```

```
  noise0=2*M_PI*rand()/(float)RAND_MAX; //Direction of noise.

  float atemp,noisetemp,proptemp,kconftemp,gtemp;
  if (rcvsnn==0)
   istrsrc(grad, noff, pcore, neighnum1[i],ringn(i,x,y,COMx,
      COMy),COMy,y[i],&atemp,&noisetemp,&proptemp,&kconftemp,&
      gtemp);  //Set all the interaction strengths for this
      particle.
  else
   istrsnn(grad, noff, pcore, neighnum1[i],ringn(i,x,y,COMx,
      COMy),COMy,y[i],&atemp,&noisetemp,&proptemp,&kconftemp,&
      gtemp);

  float v=sqrtf(vx[i]*vx[i]+vy[i]*vy[i]);   //vvvvv
  if (v==0)
   v=1;
  float nx=vx[i]/v+atemp*Ax+gtemp*gradx, ny=vy[i]/v+atemp*Ay+
     gtemp*grady;  //Propulsion vector.
  float nnorm=sqrtf(nx*nx+ny*ny);    //^^^^^
  if (nnorm==0)
   nnorm=1;

  vx[i]=proptemp*nx/nnorm+noisetemp*cosf(noise0)+e*LJx-kspring
     *Springx-kconftemp*Confx;
  vy[i]=proptemp*ny/nnorm+noisetemp*sinf(noise0)+e*LJy-kspring
     *Springy-kconftemp*Confy;
  v=sqrtf(vx[i]*vx[i]+vy[i]*vy[i]);
 }
 free(angold);
 return;
}
```

A.2.14 intermags.cpp

This program finds the final gradient interaction.

```
#include <iostream>

void istrsrc(float grad, float noff, float pcore, int neighnum
   , int ring, float COMy, float y, float *atemp, float *
   noisetemp, float *proptemp, float *kconftemp, float *gtemp)
   {
 *atemp=aoff-aslope*(6-(neighnum-1));
 *noisetemp=noff-nslope*(6-(neighnum-1));
 float conc=y-COMy;
 if (conc>L/2)
  conc-=L;
 if (conc<-L/2)
  conc+=L;

 if (ring>=confring){
  *proptemp=prim+conc*grad;
  *kconftemp=kconfine;
  *gtemp=gdep;
//  if (coup==1)
//   *kconftemp=0;
 }
 else {
  *proptemp=pcore+conc*grad;
  *gtemp=gdep;
  *kconftemp=0;
 }
 return;
}

void istrsnn(float grad, float noff, float pcore, int neighnum
   , int ring, float COMy, float y, float *atemp, float *
   noisetemp, float *proptemp, float *kconftemp, float *gtemp)
   {
```

```
 float conc=y-COMy+gdep;
 if (conc>L/2)
  conc-=L;
 if (conc<-L/2)
  conc+=L;
 *atemp=aoff-aslope*(6-(neighnum-1));
 *noisetemp=noff-nslope*(6-(neighnum-1));
 *proptemp=pcore+3.0/7.0*(pcore-prim)*(neighnum-7.0);


 *gtemp=(conc)*grad;
 *kconftemp=0;
 if (ring==confring)
  *kconftemp=kconfine;
 return;
}
```

A.2.15 LJ.cpp

This program has the Lennard-Jones interactions.

```
#include <iostream>
#include <math.h>

using namespace std;

void LJ(int i, int j, float dx, float dy, float d, float ri[],
    float *LJx, float *LJy){
 float avgr=(ri[i]+ri[j])/2.0;
 if (powf(d,13.0)<0.000001&&i!=j)
  d=0.34551;
 if (i!=j){
  *LJx+=-12*(powf((avgr),12.0)/powf((d),13.0)-powf((avgr),6.0)
     /powf((d),7.0))*dx/(d);
  *LJy+=-12*(powf((avgr),12.0)/powf((d),13.0)-powf((avgr),6.0)
     /powf((d),7.0))*dy/(d);
```

```
 }
 return;
}
```

A.2.16 main.cpp

This program is the main script that runs the others.

```
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <fstream>
#include <vector>
#include <numeric>
#include <random>
#include <chrono>
#include <ctime>

//#define N 37   //Number of Cells.
#define confring 3
#define aoff 6  //Maximum value of alpha (alignment
   interaction strength) occurs for particles with no
   neighbors.
#define rcvsnn 1  //0 for rim/core dependence, 1 for nn
   dependence
#define prim 8  //Rim p. (or slope).
#define kconfine 0
#define rs 3
#define r 30   //Cell diameter (position of Lennard-Jones well
   )
#define gdep 100
#define vini 1
#define R1 38   //Distance cut off for first nearest neighbors
   .
#define R2 100   //Distance cut off to second nearest
   neighbors.
```

```
#define kspring 0.1  //Magnitude of the cohesive spring force.
#define aslope 0  //Minimum value of alpha, occurs for
   particles with maximum number of neighbors.
#define e 18   //Lennard-Jones interaction strength. (Well
   depth)
#define nslope 0  //Minimum value of the noise. (See alpha
   description)
#define timestep 0.01  //Simulation timestep
#define samples 1  //Number of samples
#define L 1900   //Periodic boundary box edge lenghth.
#define _USE_MATH_DEFINES

#include "distx.cpp"
#include "distance.cpp"
#include "COM.cpp"
#include "ringnum.cpp"
#include "initializecircle.cpp"
#include "gradient.cpp"
#include "interactions.cpp"
#include "move.cpp"
#include "opramt.cpp"
#include "ropramt.cpp"
#include "velcor.cpp"
#include "ringrot.cpp"
//#include "clustering.cpp"
#include "avgvels.cpp"
//#include "ringorder.cpp"
#include "neighborlist.cpp"
#include "rimcoreexchange.cpp"

using namespace std;


int main(int argc, char *argv[]){
 srand (time(NULL));
 int i, j, s, tt;
```

```
float t, t1=clock();
int N = atoi(argv[1]);
float pcore = atof(argv[2]);
float noff = atof(argv[3]);
float grad = atof(argv[4]);
int T = atoi(argv[5]);
//Particle properties, things that are specific for each cell
   :
float x[N], y[N], vx[N], vy[N], ri[N];

//Arrays needed for analysis.
int clusters, label[N], rimorcore[N], rimorcoreOld[N];
float order, rorder, COMv, COMx, COMy, vavg, xold[N], yold[N
   ], cenor, cenror, cenCOMv, cenavgv, rimor, rimror, rimCOMv
   , rimavgv, rot[100], exchanges=0, count=0;

//Some System-wide properties.
int **neigh1, **neigh2, **neighold, neighnum1[N], neighnum2[N
   ], neighnumold[N];
neigh1=(int **)malloc(N*sizeof(int*));
neigh2=(int **)malloc(N*sizeof(int*));
neighold=(int **)malloc(N*sizeof(int*));



ofstream f0x;
f0x.open("runtime.txt");

ofstream f0y;
f0y.open("ri.txt");


for (s=0; s<samples; s++){  //Loop through the number of
   samples.
 //Files.
 FILE * f00; //Parameters file.
```

```
f00 = fopen ("params.txt", "w");
fprintf(f00, " N =  %d \n L =  %d \n r =  %f \n e =  %f \n
   ao =  %f \n as = %f \n no = %f \n ns = %f \n samples =  %
   d \n seed =  %ld", N, L, (float)r, (float)e, (float)aoff,
    (float)aslope, (float)noff, (float)nslope, samples, time
   (NULL));

ofstream f01; //For videos with VMD, only writes when
   samples=1.
f01.open("pos.xyz");

FILE * f01a;
//FILE * f01b;



FILE * f04;
ofstream f02; //The order parameter of the system as a
   function of time.
f02.open("data/ordervst.dat");

ofstream f03; //Rotational order/angular momentum of the
   system with t.
f03.open("data/rordervst.dat");

ofstream f05; //The velocity of the center of mass of the
   cluster of cells vs time.
f05.open("data/velocityvst.dat");

ofstream f06;
f06.open("data/rotations.dat");

ofstream f09;
f09.open("data/navg.dat");

ofstream f10;
```

```
f10.open("data/avgv.dat");

ofstream f11;
f11.open("data/rimcoreexchange.dat");

FILE *FName[T];
// Make N text files, each file would be for one agents
   trajectory through out the simulation (e.g. Agent0001.txt
    will be for Agent 1)
for (int i = 0; i < T; i++){
 if (i%4 == 0 && i>4){
  char File_num[32];
  snprintf(File_num, sizeof(char)*32, "./agents_position/T
     %05d.txt", i);
  FName[i] = fopen(File_num, "w");
 }

}

//Initialize particles (perfect packed hexagon, with random
   directions, and velocities).
initialize(N,x, y, vx, vy, xold, yold, ri, rimorcore,
   rimorcoreOld);
for (i=0; i<N; i++)
 neighnumold[i]=0;
for (tt=0; tt<(float)T/(float)timestep; tt++){  //Loop
   through number of time steps.
 t=tt*timestep;  //Actual time in minutes.
 com(0,N-1,x,y,&COMx,&COMy);
 if (samples==1&&tt%50==0){ //Make vmd video file.
  f01 << N << "\n\n";
  for (i=0; i<N; i++)
   f01 << "H " << x[i] << " " << y[i] << " 0.0\n";

  char buffer1[32];
  //char buffer2[32];
```

```
 snprintf(buffer1, sizeof(char)*32, "./video/fr%06d", tt);
 //snprintf(buffer2, sizeof(char)*32, "agent_files/Tpoint
    %06d.dat", tt);
 f01a=fopen(buffer1, "w");
 //f01b=fopen(buffer2, "w");
 for (i=0; i<N; i++){
  fprintf(f01a, "%d %f %f %f\n", rimorcore[i], x[i], y[i],
     ri[i]);
  //fprintf(f01b, "%d %d %f %f %f %f\n", rimorcore[i],
     rimorcoreOld[i], x[i], y[i], xold[i], yold[i]);
 }
 fclose(f01a);
 //fclose(f01b);
}

int coup=1; // Rim and core are coupled.

//Create neighborlists.
if (tt%1==0){
 for (i=0;i<N;i++){
  free(neigh1[i]);
  free(neigh2[i]);
  nlist(N, i,x,y,ri,neigh1,neighnum1,neigh2,neighnum2);
 }
}
if (tt==2){
 float dummy=rimcoreexchange(N, rimorcore,rimorcoreOld,
    neigh1,neigh2,neighnum1,neighnum2,x,y);
}

//Update directions.
interactions(N, grad, noff, pcore, x, y, COMx, COMy, vx, vy
   , neigh1, neigh2, neighnum1, neighnum2, coup, ri); //
   This function edits caculates forces on each particle
   and updates vx, and vy, accordingly.
```

```
   //Update positions.  Updates the x and y coordinates of
      each cell based on the vx, and vy, calculated before.
   move(N, x, y, vx, vy);
   exchanges=rimcoreexchange(N, rimorcore,rimorcoreOld, neigh1
      ,neigh2,neighnum1,neighnum2,x,y);
   if (tt%400 == 0 && tt>401){
    int IND = t;
    for (int jj = 0; jj < N; jj++){
     fprintf(FName[IND], "%f %f %d %f %f %d\n", xold[jj], yold
        [jj], rimorcoreOld[jj], x[jj], y[jj], rimorcore[jj]);
    }
    // Closes all the agent files
    fclose(FName[IND]);
   }


   //Find the velocity of the center of mass of the cell
      cluster.

   if (tt%400==0&&tt>401){    //Everyone second timestep write
       the order, angular momentum and velocities as well as
      time into the appropriate files. Do analysis and enter
      numbers into files.
//    if (tt>T/(float)timestep-527){
//     velcor(Num,L,x,y,xold,yold,vx,vy);
//    }

    ringrot(N,x,y,xold,yold,COMx,COMy,rot);
    avgvs(N, x,y,xold,yold,&COMv,&vavg);
    order = opram(N, vx, vy, xold, yold, x, y);
    rorder = ropram(N, x, y, vx, vy, xold, yold, rimorcore,
       rimorcoreOld);  //Note that xold and yold get
       overwritten to new x and y in this function(ropram).
    //exchanges=rimcoreexchange(N, rimorcore,rimorcoreOld,
       neigh1,neigh2,neighnum1,neighnum2,x,y);
```

```
    f02 << t+s*T << " " << order << endl;
    f03 << t+s*T << " " << rorder << endl;
    f05 << t+s*T << " " << COMv << endl;
    f06 << t+s*T;
    for (int i=1; i<confring+1; i++)
     f06 << " " << rot[i];
    f06 << endl;
    f10 << t+s*T << " " << vavg << endl;
    f11 << t+s*T << " " << exchanges << endl;

   }
  }
 }
 float t2=clock();
 f0x << (t2-t1)/CLOCKS_PER_SEC << endl;
 return 0;
}
```

A.2.17 move.cpp

This program move the cells based on velocities.

```
#include <iostream>
#include <math.h>

void move(int N, float x[], float y[], float vx[], float vy[])
   {
 int i;
 for (i=0; i<N; i++){
  x[i]+=vx[i]*timestep;
  y[i]+=vy[i]*timestep;
  x[i]=fmod(x[i]+L,L);
  y[i]=fmod(y[i]+L,L);
 }
 return;
}
```

A.2.18 neighborlist.cpp

This program finds cells nearest neighbors.

```
#include <iostream>

void nlist(int N, int i, float x[], float y[], float ri[], int
   * neigh1[], int neighnum1[], int* neigh2[], int neighnum2
   []){
 int *temp1=(int*)malloc(1*sizeof(int)), *new_neigh1;
 int *temp2=(int*)malloc(1*sizeof(int)), *new_neigh2;

 neighnum1[i]=0;
 neighnum2[i]=0;
 float d, dx, dy;
 for (int j=0; j<N; j++){
  d=dist(i,j,x,y,&dx, &dy);
  if (d<(ri[i]+ri[j])/2.0+R1-r){
   neighnum1[i]++;
   new_neigh1=(int*)realloc(temp1,neighnum1[i]*sizeof(int));
   if (new_neigh1==NULL){
    free(temp1);
    break;
   }
   else{
    temp1=new_neigh1;
    temp1[neighnum1[i]-1]=j;
   }
  }
  if (d>(ri[i]+ri[j])/2.0+R1-r && d<(ri[i]+ri[j])/2.0+R2-r){
   neighnum2[i]++;
   new_neigh2=(int*)realloc(temp2,neighnum2[i]*sizeof(int));
   if (new_neigh2==NULL){
    free(temp2);
    break;
   }
   else{
```

```
    temp2=new_neigh2;
    temp2[neighnum2[i]-1]=j;
   }
  }
 }
 neigh1[i]=temp1;
 neigh2[i]=temp2;
 return;
}
```

A.2.19 opramt.cpp

This program finds the order parameter.

```
#include <iostream>
#include <math.h>
float opram(int N, float vx[], float vy[], float xold[], float
    yold[], float x[], float y[]){
 float txvel=0, tyvel=0, v;
 int i;
 for (i=0; i<N; i++){
  v=sqrtf((x[i]-xold[i])*(x[i]-xold[i])+(y[i]-yold[i])*(y[i]-
     yold[i]));
  if (v>0){
   txvel+=(x[i]-xold[i])/v;
   tyvel+=(y[i]-yold[i])/v;
  }
 }
 float order=sqrtf(txvel*txvel+tyvel*tyvel)/((float)N);
 return (order);
}
```

A.2.20 rimcoreexchange.cpp

This program finds the exchanges. if it went from rim to core or core to rim.

```
#include <iostream>
```

```
#include <stdlib.h>
#include <math.h>

float rimcoreexchange(int N, int rimorcore[], int rimorcoreOld
   [], int **neigh1, int **neigh2, int *neighnum1, int *
   neighnum2, float x[], float y[]){
 float gradx=0,grady=0;
 int exchanges=0;
 for (int i=0; i<N; i++){
  gradx=0;
  grady=0;
  gradient(i, neigh1[i], neigh2[i], neighnum1[i], neighnum2[i
     ], x, y, &gradx, &grady);
  if (sqrtf(gradx*gradx+grady*grady)<M_PI/18.0 && rimorcore[i
     ]==1){

   rimorcore[i]=0;
   exchanges++;
  }
  if (sqrtf(gradx*gradx+grady*grady)>M_PI/18.0 && rimorcore[i
     ]==0){

   rimorcore[i]=1;
   exchanges++;
  }
 }
 return (float)exchanges;
}
```

A.2.21 ringnum.cpp

This program finds normalized distance of cells from center of mass.

```
#include <iostream>
#include <math.h>
```

```
int ringn(int i, float x[], float y[], float COMx, float COMy)
   {
 float COMxtemp=COMx, COMytemp=COMy;
        if ((x[i]-COMx)>L/2)
                COMxtemp+=L;
        if ((x[i]-COMx)<-L/2)
                COMxtemp-=L;
        if ((y[i]-COMy)>L/2)
                COMytemp+=L;
        if ((y[i]-COMy)<-L/2)
                COMytemp-=L;
        float rad=sqrtf((x[i]-COMxtemp)*(x[i]-COMxtemp)+(y[i]-
           COMytemp)*(y[i]-COMytemp));
//         if (rad==0)
//                 rad=0.00001;
// int ring=1, checking=i;
// if (i==0)
//  ring=0;
// for (int rin=0;rin<ring;rin++){
//  if (checking-(rin+1)*6>0){
//   ring++;
//   checking-=(rin+1)*6;
//  }
// }
 return round(rad/r);
}
```

A.2.22 ringorder.cpp

This program calculates desired order for desired rings.

```
#include <iostream>
```

```
//Inputs for this function to calculate desired order for
   desired rings: Total number of particles, lowest desired
   ring number, total number or rings desired, 0 for
   polarization or 1 for angular momentum,
void ringorder(int firstring, int ringnum, int rvsopram, float
    x[], float y[], float vx[], float vy[], float xold[],
   float yold[], float *order, float *rorder, float *COMv,
   float *vavg) {
 float *newx, *newy, *newvx, *newvy, *newxold, *newyold;
 int size=0, firsti=0;
 for (int i=firstring; i<firstring+ringnum; i++){
  if (i==0)
   size++;
  size+=i*6;
 }
 for (int i=0; i<firstring;i++){
  if (i==0)
   firsti++;
  firsti+=i*6;
 }

 newx=(float*)malloc(size*sizeof(float));
 newy=(float*)malloc(size*sizeof(float));
 newvx=(float*)malloc(size*sizeof(float));
 newvy=(float*)malloc(size*sizeof(float));
 newxold=(float*)malloc(size*sizeof(float));
 newyold=(float*)malloc(size*sizeof(float));

 for (int i=0; i<size; i++){
  newx[i]=x[i+firsti];
  newy[i]=y[i+firsti];
  newvx[i]=vx[i+firsti];
  newvy[i]=vy[i+firsti];
  newxold[i]=xold[i+firsti];
  newyold[i]=yold[i+firsti];
 }
```

```
 avgvs(size,newx,newy,newxold,newyold,COMv,vavg);
 *order=opram(newvx,newvy,size,newxold,newyold,newx,newy);
 *rorder=ropram(newx,newy,newvx,newvy,size,L,newxold,newyold);

 return;
}
```

A.2.23 ringrot.cpp

This program finds angular momentum for the ring/

```
#include <iostream>

void ringrot(int N, float x[], float y[], float xold[], float
   yold[], float COMx, float COMy, float rot[]){

 float COMxold, COMyold, rxold, ryold, rxnew, rynew;
 int size[100];

 com(0,N-1,xold,yold,&COMxold,&COMyold);

 for (int i=0;i<confring+1;i++){
  rot[i]=0;
  size[i]=0;
 }

 for (int i=0;i<N;i++){
  rxold=distx(xold[i],COMxold);
  ryold=distx(yold[i],COMyold);
  rxnew=distx(x[i],COMx);
  rynew=distx(y[i],COMy);
  rot[ringn(i,x,y,COMx,COMy)]+= 2.*M_PI/(float)L * distx( (
     float)L/(2.*M_PI) * atan2f(ryold,rxold) , (float)L/(2.*
     M_PI)*atan2f(rynew,rxnew) );
  size[ringn(i,x,y,COMx,COMy)]++;
 }
```

```
 for (int i=0;i<confring+1;i++){
  rot[i]=rot[i]/(float)size[i];
 }



 return;
}
```

A.2.24 ropramt.cpp

This program finds angular momentum for entire cluster.

```
#include <iostream>
#include <math.h>
float ropram(int N, float x[], float y[], float vx[], float vy
   [], float xold[], float yold[], int rimorcore[], int
   rimorcoreOld[]){
 float COMx=0, COMy=0, rorder=0, k=0, xx, yy;
 int i;
 //Find Center of mass.
 for (i=0; i<N; i++){
  k++;
  xx=x[i];
  yy=y[i];
  if (COMx-xx>L/2)
   xx+=L;
  if (COMx-xx<-L/2)
   xx-=L;
                if (COMy-yy>L/2)
                        yy+=L;
                if (COMy-yy<-L/2)
                        yy-=L;
//  if (sqrtf((xx-COMx)*(xx-COMx)+(yy-COMy)*(yy-COMy))<sqrtf((
   float)N)){
   COMx = COMx + (xx - COMx)/(float)k;
```

```
   COMy = COMy + (yy - COMy)/(float)k;
//  }
//  else
//   k=k-1;
 }
 if (COMx<0)
  COMx+=L;
 if (COMy<0)
  COMy+=L;

 //Calculate rotational order parameter.
 for (i=0; i<N; i++){
  float rx=x[i]-COMx;
  if (rx>L/2)
   rx-=L;
  else if (rx<-L/2)
   rx+=L;

  float ry=y[i]-COMy;
  if (ry>L/2)
                        ry-=L;
               else if (ry<-L/2)
                        ry+=L;

  float rt=sqrtf(rx*rx+ry*ry);
  float v=sqrtf((x[i]-xold[i])*(x[i]-xold[i])+(y[i]-yold[i])*(
     y[i]-yold[i]));


  rorder+=((x[i]-xold[i])*ry-(y[i]-yold[i])*rx)/(rt*v);


 }
 for (i=0;i<N;i++){
  xold[i]=x[i];
  yold[i]=y[i];
```

```
  rimorcoreOld[i] = rimorcore[i];
 }
 rorder=rorder/(float)N;
 if (rorder<0)
  rorder=-rorder;
 return (rorder);
}
```

A.2.25 spring.cpp

This program calculates spring interaction strength between next nearest neighbors.

```
#include <iostream>
#include <math.h>

void spr(int i, int j, float x[], float y[], int neighnum, int
    neigh[], float *Springx, float *Springy){
 int kspringtemp=1;
 float dx2,dy2,dx,dy;
 float d2=dist(i,j,x,y,&dx2,&dy2);
 for (int ij=0;ij<neighnum;ij++){
  float d=dist(i,neigh[ij],x,y,&dx,&dy);
  float sdotd=(dx2)*(dx)+(dy2)*(dy);
  if (sqrtf((d*d2)*(d*d2)-powf((dx*dx2+dy*dy2),2))/d2<r&&sdotd
     >0)
   kspringtemp=0;
 }
 *Springx-=kspringtemp*(dx2);
 *Springy-=kspringtemp*(dy2);
 return;
}
```

A.2.26 velcor.cpp

This program finds velocity velocity correlation.

```
#include <iostream>
```

```
#include <stdlib.h>
#include <math.h>
#include <fstream>
//FARNAZ NOTE: THIS FINDS CORRELATION FUNCTION

using namespace std;

void velcor(int N, float x[], float y[], float xold[], float
   yold[], float vx[], float vy[]){
 ofstream f00;
 f00.open("data/velcor.dat");

 int i, j;
 float d, dx, dy;
 for (i=0; i<N-1; i++){
  float vi=sqrtf((x[i]-xold[i])*(x[i]-xold[i])+(y[i]-yold[i])
     *(y[i]-yold[i]));
  for (j=i+1; j<N; j++){
   float vj=sqrtf((x[j]-xold[j])*(x[j]-xold[j])+(y[j]-yold[j])
      *(y[j]-yold[j]));
   dx=x[j]-x[i];
                        if (dx>L/2)
                                dx-=L;
                        if (dx<-L/2)
                                dx+=L;

                        dy=y[j]-y[i];
                        if (dy>L/2)
                                dy-=L;
                        if (dy<-L/2)
                                dy+=L;

                        d=sqrtf(dx*dx+dy*dy);

   f00 << d << " " << ((x[i]-xold[i])*(x[j]-xold[j])+(y[i]-
      yold[i])*(y[j]-yold[j]))/(vi*vj) << endl;
```

```
  }
 }
 return;
}
```

A.2.27 velocity.cpp

This program calculates velocity based on number of nearest neighbors.

```
#include <iostream>

void velocity(int N, float L, float v_max, float v_min, float
   v_noise, float COMnoise, float x[], float y[], float v[],
   int label[], int cls, int* neigh[], float* COMx, float*
   COMy, int neighnum){
/*************************************************************
CALCULATE V BASED ON DISTANCE FROM CLUSTER COM
*************************************************************/
/*
 int i, c, *k;
 float *r, *rmax, xi, yi;
 r=(float *)malloc(N*sizeof(float));
 rmax=(float *)malloc(cls*sizeof(float));
 k=(int *)malloc(cls*sizeof(int));

 for (c=0; c<cls; c++){
  COMx[c]=0;
  COMy[c]=0;
  rmax[c]=0;
  k[c]=0;
 }

 for (i=0; i<N; i++){
  k[label[i]]++;
  r[i]=0;
  xi=x[i];
```

```
 yi=y[i];

 if (COMx[label[i]]-xi>L/2)
  xi+=L;
 else if (COMx[label[i]]-xi<-L/2)
  xi-=L;
 if (COMy[label[i]]-yi>L/2)
  yi+=L;
 else if (COMy[label[i]]-yi<-L/2)
  yi-=L;

 COMx[label[i]] = COMx[label[i]]+(xi-COMx[label[i]])/(float)k
    [label[i]];
 COMy[label[i]] = COMy[label[i]]+(yi-COMy[label[i]])/(float)k
    [label[i]];
}

for (c=0; c<cls; c++){
 if (COMx[c]<0)
  COMx[c]+=L;
 if (COMy[c]<0)
  COMy[c]+=L;
}

for (i=0; i<N; i++){
 float dx=x[i]-COMx[label[i]];
 if (dx>L/2)
  dx-=L;
 if (dx<-L/2)
  dx+=L;

 float dy=y[i]-COMy[label[i]];
 if (dy>L/2)
  dy-=L;
 if (dy<-L/2)
  dy+=L;
```

```
  r[i]=sqrt(dx*dx+dy*dy);
  if (r[i]>rmax[label[i]])
   rmax[label[i]]=r[i];
 }
 for (i=0; i<N; i++){
  if (rmax[label[i]]!=0){
   v[i]=1*((v_max-v_min)*r[i]/rmax[label[i]]+v_min+v_noise*(
      float)rand()/(float)RAND_MAX-v[i])+v[i];
  }
  else {
   v[i]=1*(v_max-v[i]) + v[i];
  }
 }
 free(r);
 free(rmax);
 free(k);
*/
/*****************************************************
CALCULATE V FROM # OF NEAREST NEIGHBORS
*****************************************************/

 int i, j;
 for (i=0; i<N; i++){
  j=0;
  do {
   j++;
  } while (neigh[i][j]!=-1);
  v[i]=/*0.005*(float)rand()/(float)RAND_MAX**/(neighnum+1-j)*
     v_max/neighnum;
 }

 return;
}
```

Bibliography

[1] Gandhimohan M Viswanathan, Marcos GE Da Luz, Ernesto P Raposo, and H Eugene Stanley. *The physics of foraging: an introduction to random searches and biological encounters*. Cambridge University Press, 2011.

[2] Gandimohan M Viswanathan, Sergey V Buldyrev, Shlomo Havlin, MGE Da Luz, EP Raposo, and H Eugene Stanley. Optimizing the success of random searches. *Nature*, 401(6756):911, 1999.

[3] Tamás Vicsek and Anna Zafeiris. Collective motion. *Physics Reports*, 517(3-4):71–140, 2012.

[4] Elena Scarpa and Roberto Mayor. Collective cell migration in development. *Journal of Cell Biology*, 212(2):143–155, 2016.

[5] Andrew M Reynolds, Jennifer L Swain, Alan D Smith, Andrew P Martin, and Juliet L Osborne. Honeybees use a lévy flight search strategy and odour-mediated anemotaxis to relocate food sources. *Behavioural Ecology and Sociobiology*, 64(1):115, 2009.

[6] RPD Atkinson, CJ Rhodes, DW Macdonald, and RM Anderson. Scale-free dynamics in the movement patterns of jackals. *Oikos*, 98(1):134–140, 2002.

[7] Gabriel Ramos-Fernández, José L Mateos, Octavio Miramontes, Germinal Cocho, Hernán Larralde, and Barbara Ayala-Orozco. Lévy walk patterns in the foraging movements of spider monkeys (ateles geoffroyi). *Behavioural Ecology and Sociobiology*, 55(3):223–230, 2004.

[8] Gandhimohan M Viswanathan, V Afanasyev, SV Buldyrev, EJ Murphy, PA Prince, and H Eugene Stanley. Lévy flight search patterns of wandering albatrosses. *Nature*, 381(6581):413, 1996.

[9] Julius Adler. Chemotaxis in bacteria. *Science*, 153(3737):708–716, 1966.

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

[11] Frederic Bartumeus and Jordi Catalan. Optimal search behaviour and classic foraging theory. *Journal of Physics A: Mathematical and Theoretical*, 42(43):434002, 2009.

[12] Toby A Patterson, Len Thomas, Chris Wilcox, Otso Ovaskainen, and Jason Matthiopoulos. State–space models of individual animal movement. *Trends in Ecology & Evolution*, 23(2):87–94, 2008.

[13] Mathieu Lihoreau, Michael A Charleston, Alistair M Senior, Fiona J Clissold, David Raubenheimer, Stephen J Simpson, and Jerome Buhl. Collective foraging in spatially complex nutritional environments. *Phil. Trans. R. Soc. B*, 372(1727):20160238, 2017.

[14] Kristina Lerman and Aram Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2):127–141, 2002.

[15] Esa Ranta, Hannu Rita, and Nina Peuhkuri. Patch exploitation, group foraging, and unequal competitors. *Behavioural Ecology*, 6(1):1–5, 1995.

[16] Blazej Bulka, Matthew Gaston, and Marie Desjardins. Local strategy learning in networked multi-agent team formation. *Autonomous Agents and Multi-Agent Systems*, 15(1):29–45, 2007.

[17] PY Quenette, J Ferron, and L Sirois. Group foraging in snowshoe hares (lepus americanus): Aggregation or social group? *Behavioural Processes*, 41(1):29–37, 1997.

[18] GM Viswanathan, V Afanasyev, Sergey V Buldyrev, Shlomo Havlin, MGE Da Luz, EP Raposo, and H Eugene Stanley. Lévy flights in random searches. *Physica A: Statistical Mechanics and its Applications*, 282(1-2):1–12, 2000.

[19] GM Viswanathan, Frederic Bartumeus, Sergey V Buldyrev, Jordi Catalan, UL Fulco, Shlomo Havlin, MGE Da Luz, Marcelo Leite Lyra, EP Raposo, and

H Eugene Stanley. Lévy flight random searches in biological phenomena. *Physica A: Statistical Mechanics and its Applications*, 314(1-4):208–213, 2002.

[20] Katherine Copenhagen, Gema Malet-Engra, Weimiao Yu, Giorgio Scita, Nir Gov, and Ajay Gopinathan. Frustration-induced phases in migrating cell clusters. *Science Advances*, 4(9):eaar8483, 2018.

[21] Shlomo Havlin and Daniel Ben-Avraham. Diffusion in disordered media. *Advances in Physics*, 36(6):695–798, 1987.

[22] HE Hurst, RP Black, and YM Simaika. Long-term storage: An experimental study (london: Constable). 1965.

[23] A-L Barabási, Harry Eugene Stanley, et al. *Fractal concepts in surface growth*. Cambridge University Press, 1995.

[24] IM Sokolov. Lévy flights from a continuous-time process. *Physical Review E*, 63(1):011104, 2000.

[25] R Metzler and IM Sokolov. Superdiffusive klein-kramers equation: Normal and ano malous time evolution and lévy walk moments. *EPL (Europhysics Letters)*, 58(4):482, 2002.

[26] Yasuhiro Matsunaga, Chun-Biu Li, and Tamiki Komatsuzaki. Anomalous diffusion in folding dynamics of minimalist protein landscape. *Physical Review Letters*, 99(23):238103, 2007.

[27] A Ao Tsonis, PJ Roebber, and JB Elsner. A characteristic time scale in the global temperature record. *Geophysical Research Letters*, 25(15):2821–2823, 1998.

[28] AA Tsonis, AG Hunt, and JB Elsner. On the relation between enso and global climate change. *Meteorology and Atmospheric Physics*, 84(3):229–242, 2003.

[29] Markus Porto and H Eduardo Roman. Autoregressive processes with exponentially decaying probability distribution functions: Applications to daily variations of a stock market index. *Physical Review E*, 65(4):046149, 2002.

[30] Harry Eugene Stanley, Luis A Nunes Amaral, David Canning, Parameswaran Gopikrishnan, Youngki Lee, and Yanhui Liu. Econophysics: Can physicists contribute to the science of economics? *Physica A: Statistical Mechanics and its Applications*, 269(1):156–169, 1999.

[31] Luis AN Amaral, P Cizeau, P Gopikrishnan, Y Liu, M Meyer, C-K Peng, and HE Stanley. Econophysics: can statistical physics contribute to the science of economics? *Computer Physics Communications*, 121:145–152, 1999.

[32] Eduardo Pereira, José MG Martinho, and Mário N Berberan-Santos. Photon trajectories in incoherent atomic radiation trapping as lévy flights. *Physical Review Letters*, 93(12):120201, 2004.

[33] Diederik S Wiersma. The physics and applications of random lasers. *Nature Physics*, 4(5):359–367, 2008.

[34] Koyo Tamura, Yusril Yusuf, Yoshiki Hidaka, and Shoichi Kai. Nonlinear transport and anomalous brownian motion in soft-mode turbulence. *Journal of the Physical Society of Japan*, 70(10):2805–2808, 2001.

[35] SV Buldyrev, S Havlin, A Ya Kazakov, MGE Da Luz, EP Raposo, HE Stanley, and GM Viswanathan. Average time spent by lévy flights and walks on an interval with absorbing boundaries. *Physical Review E*, 64(4):041108, 2001.

[36] SV Buldyrev, M Gitterman, S Havlin, A Ya Kazakov, MGE Da Luz, EP Raposo, HE Stanley, and GM Viswanathan. Properties of lévy flights on an interval with absorbing boundaries. *Physica A: Statistical Mechanics and its Applications*, 302(1-4):148–161, 2001.

[37] Kolbjørn Tunstrøm, Yael Katz, Christos C Ioannou, Cristián Huepe, Matthew J Lutz, and Iain D Couzin. Collective states, multistability and transitional behaviour in schooling fish. *PLoS Computational Biology*, 9(2):e1002915, 2013.

[38] William Bialek, Andrea Cavagna, Irene Giardina, Thierry Mora, Oliver Pohl, Edmondo Silvestri, Massimiliano Viale, and Aleksandra M Walczak. Social interactions dominate speed control in poising natural flocks near criticality. *Proceedings of the National Academy of Sciences*, 111(20):7212–7217, 2014.

[39] James A Shapiro. Thinking about bacterial populations as multicellular organisms. *Annual Review of Microbiology*, 52(1):81–104, 1998.

[40] Volker Schaller, Christoph Weber, Christine Semmrich, Erwin Frey, and Andreas R Bausch. Polar patterns of driven filaments. *Nature*, 467(7311):73–77, 2010.

[41] Douglas H Kelley and Nicholas T Ouellette. Emergent dynamics of laboratory insect swarms. *Scientific Reports*, 3(1):1–7, 2013.

[42] Tamas Vicsek. A question of scale. *Nature*, 411(6836):421–421, 2001.

[43] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226, 1995.

[44] Guillaume Grégoire and Hugues Chaté. Onset of collective and cohesive motion. *Physical Review Letters*, 92(2):025702, 2004.

[45] Peter Friedl and Darren Gilmour. Collective cell migration in morphogenesis, regeneration and cancer. *Nature Reviews Molecular Cell Biology*, 10(7):445, 2009.

[46] Mathieu Poujade, Erwan Grasland-Mongrain, A Hertzog, J Jouanneau, Philippe Chavrier, Benoît Ladoux, Axel Buguin, and Pascal Silberzan. Collective migration of an epithelial monolayer in response to a model wound. *Proceedings of the National Academy of Sciences*, 104(41):15988–15993, 2007.

[47] Peter Friedl, Yael Hegerfeldt, and Miriam Tusch. Collective cell migration in morphogenesis and cancer. *International Journal of Developmental Biology*, 48(5-6):441–449, 2004.

[48] Andy Aman and Tatjana Piotrowski. Cell migration during morphogenesis. *Developmental Biology*, 341(1):20–33, 2010.

[49] Kevin J Cheung and Andrew J Ewald. A collective route to metastasis: Seeding by tumor cell clusters. *Science*, 352(6282):167–169, 2016.

[50] Christina H Stuelten, Carole A Parent, and Denise J Montell. Cell motility in cancer invasion and metastasis: insights from simple model organisms. *Nature Reviews Cancer*, 18(5):296–312, 2018.

[51] Roberto Mayor and Sandrine Etienne-Manneville. The front and rear of collective cell migration. *Nature Reviews Molecular Cell Biology*, 17(2):97–109, 2016.

[52] Thomas D Pollard and John A Cooper. Actin, a central player in cell shape and movement. *Science*, 326(5957):1208–1212, 2009.

[53] Hui Xu, Ding Ye, Martine Behra, Shawn Burgess, Songhai Chen, and Fang Lin. Gβ1 controls collective cell migration by regulating the protrusive activity of leader cells in the posterior lateral line primordium. *Developmental Biology*, 385(2):316–327, 2014.

[54] Mark A Barber and Heidi CE Welch. Pi3k and rac signalling in leukocyte and cancer cell migration. *Bulletin du Cancer*, 93(5):10044–10052, 2006.

[55] Wenjun Guo and Filippo G Giancotti. Integrin signalling during tumour progression. *Nature Reviews Molecular Cell Biology*, 5(10):816–826, 2004.

[56] Bo Shen, M Keegan Delaney, and Xiaoping Du. Inside-out, outside-in, and inside–outside-in: G protein signaling in integrin-mediated cell adhesion, spreading, and retraction. *Current Opinion in Cell Biology*, 24(5):600–606, 2012.

[57] Denise J Montell, Wan Hee Yoon, and Michelle Starz-Gaiano. Group choreography: mechanisms orchestrating the collective movement of border cells. *Nature Reviews Molecular Cell Biology*, 13(10):631–645, 2012.

[58] Amin S Ghabrial and Mark A Krasnow. Social interactions among epithelial cells during tracheal branching morphogenesis. *Nature*, 441(7094):746–749, 2006.

[59] Shirin M Pocha and Denise J Montell. Cellular and molecular mechanisms of single and collective cell migrations in drosophila: themes and variations. *Annual Review of Genetics*, 48:295–318, 2014.

[60] Satoshi Arima, Koichi Nishiyama, Toshiyuki Ko, Yuichiro Arima, Yuji Hakozaki, Kei Sugihara, Hiroaki Koseki, Yasunobu Uchijima, Yukiko Kurihara, and Hiroki Kurihara. Angiogenic morphogenesis driven by dynamic and heterogeneous collective endothelial cell movement. *Development*, 138(21):4763–4776, 2011.

[61] Lars Jakobsson, Claudio A Franco, Katie Bentley, Russell T Collins, Bas Ponsioen, Irene M Aspalter, Ian Rosewell, Marta Busse, Gavin Thurston, Alexander

Medvinsky, et al. Endothelial cells dynamically compete for the tip cell position during angiogenic sprouting. *Nature Cell Biology*, 12(10):943–953, 2010.

[62] Gema Malet-Engra, Weimiao Yu, Amanda Oldani, Javier Rey-Barroso, Nir S Gov, Giorgio Scita, and Loïc Dupré. Collective cell motility promotes chemotactic prowess and resistance to chemorepulsion. *Current Biology*, 25(2):242–250, 2015.

[63] Gil Ariel, Amit Rabani, Sivan Benisty, Jonathan D Partridge, Rasika M Harshey, and Avraham Be'Er. Swarming bacteria migrate by lévy walk. *Nature Communications*, 6:8396, 2015.

[64] Tajie H Harris, Edward J Banigan, David A Christian, Christoph Konradt, Elia D Tait Wojno, Kazumi Norose, Emma H Wilson, Beena John, Wolfgang Weninger, Andrew D Luster, et al. Generalized lévy walks and the role of chemokines in migration of effector cd8+ t cells. *Nature*, 486(7404):545, 2012.

[65] AS Ferreira, EP Raposo, GM Viswanathan, and MGE Da Luz. The influence of the environment on lévy random search efficiency: fractality and memory effects. *Physica A: Statistical Mechanics and its Applications*, 391(11):3234–3246, 2012.

[66] Marina E Wosniack, Marcos C Santos, Ernesto P Raposo, Gandhi M Viswanathan, and Marcos GE da Luz. The evolutionary origins of lévy walk foraging. *PLoS Computational Biology*, 13(10):e1005774, 2017.

[67] Giorgio Volpe and Giovanni Volpe. The topography of the environment alters the optimal search strategy for active particles. *Proceedings of the National Academy of Sciences*, page 201711371, 2017.

[68] Noriyuki P Tani, Alan Blatt, David A Quint, and Ajay Gopinathan. Optimal cooperative searching using purely repulsive interactions. *Journal of Theoretical Biology*, 361:159–164, 2014.

[69] Ricardo Martínez-García, Justin M Calabrese, and Cristóbal López. Optimal search in interacting populations: Gaussian jumps versus lévy flights. *Physical Review E*, 89(3):032718, 2014.

[70] Ricardo Martínez-García, Justin M Calabrese, Thomas Mueller, Kirk A Olson, and Cristóbal López. Optimizing the search for resources by sharing information: Mongolian gazelles as a case study. *Physical Review Letters*, 110(24):248106, 2013.

[71] Kunal Bhattacharya and Tamás Vicsek. Collective foraging in heterogeneous landscapes. *Journal of The Royal Society Interface*, 11(100):20140674, 2014.

[72] Mark R Ryan and James J Dinsmore. A quantitative study of the behaviour of breeding american coots. *The Auk*, 96(4):704–713, 1979.

[73] James L David Smith, CHARLES McDOUGAL, and Dale Miquelle. Scent marking in free-ranging tigers, panthera tigris. *Animal Behaviour*, 37:1–10, 1989.

[74] HJ De Knegt, GM Hengeveld, F Van Langevelde, WF De Boer, and KP Kirkman. Patch density determines movement patterns and foraging efficiency of large herbivores. *Behavioural Ecology*, 18(6):1065–1072, 2007.

[75] AM Reynolds and F Bartumeus. Optimising the success of random destructive searches: Lévy walks can outperform ballistic motions. *Journal of Theoretical Biology*, 260(1):98–103, 2009.

[76] Alex James, Michael J Plank, and Andrew M Edwards. Assessing lévy walks as models of animal foraging. *Journal of the Royal Society Interface*, 8(62):1233–1247, 2011.

[77] Ken Yoda, Naoki Tomita, Yuichi Mizutani, Akira Narita, and Yasuaki Niizuma. Spatio-temporal responses of black-tailed gulls to natural and anthropogenic food resources. *Marine Ecology Progress Series*, 466:249–259, 2012.

[78] David JT Sumpter. The principles of collective animal behaviour. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 361(1465):5–22, 2006.

[79] James E Herbert-Read, Andrea Perna, Richard P Mann, Timothy M Schaerf, David JT Sumpter, and Ashley JW Ward. Inferring the rules of interaction of shoaling fish. *Proceedings of the National Academy of Sciences*, 108(46):18726–18731, 2011.

[80] Előd Méhes and Tamas Vicsek. Collective motion of cells: from experiments to models. *Integrative Biology*, 6(9):831–854, 2014.

[81] Balint Szabo, GJ Szöllösi, B Gönci, Zs Jurányi, David Selmeczi, and Tamás Vicsek. Phase transition in the collective migration of tissue cells: experiment and model. *Physical Review E*, 74(6):061908, 2006.

[82] Zhao Cheng, Zhiyong Chen, Tamás Vicsek, Duxin Chen, and Hai-Tao Zhang. Pattern phase transitions of self-propelled particles: gases, crystals, liquids, and mills. *New Journal of Physics*, 18(10):103005, 2016.

[83] Tom Brandstätter, David B Brückner, Yu Long Han, Ricard Alert, Ming Guo, and Chase P Broedersz. Curvature induces active velocity waves in rotating multicellular spheroids. *arXiv preprint arXiv:2110.14614*, 2021.

[84] Yao-li Chuang, Maria R Dorsogna, Daniel Marthaler, Andrea L Bertozzi, and Lincoln S Chayes. State transitions and the continuum limit for a 2d interacting, self-propelled particle system. *Physica D: Nonlinear Phenomena*, 232(1):33–47, 2007.

[85] Nicola Aceto, Aditya Bardia, David T Miyamoto, Maria C Donaldson, Ben S Wittner, Joel A Spencer, Min Yu, Adam Pely, Amanda Engstrom, Huili Zhu, et al. Circulating tumor cell clusters are oligoclonal precursors of breast cancer metastasis. *Cell*, 158(5):1110–1122, 2014.

[86] Yu-Chiuan Chang, Jhen-Wei Wu, Yi-Chi Hsieh, Tzu-Han Huang, Zih-Min Liao, Yi-Shan Huang, James A Mondo, Denise Montell, and Anna C-C Jang. Rap1 negatively regulates the hippo pathway to polarize directional protrusions in collective cell migration. *Cell Reports*, 22(8):2160–2175, 2018.

[87] Xavier Trepat, Michael R Wasserman, Thomas E Angelini, Emil Millet, David A Weitz, James P Butler, and Jeffrey J Fredberg. Physical forces during collective cell migration. *Nature Physics*, 5(6):426, 2009.

[88] Dhananjay T Tambe, C Corey Hardin, Thomas E Angelini, Kavitha Rajendran, Chan Young Park, Xavier Serra-Picamal, Enhua H Zhou, Muhammad H Zaman, James P Butler, David A Weitz, et al. Collective cell guidance by cooperative intercellular forces. *Nature Materials*, 10(6):469, 2011.

[89] Carlos Carmona-Fontaine, Helen K Matthews, Sei Kuriyama, Mauricio Moreno, Graham A Dunn, Maddy Parsons, Claudio D Stern, and Roberto Mayor. Contact inhibition of locomotion in vivo controls neural crest directional migration. *Nature*, 456(7224):957, 2008.

[90] Ajay Gopinathan and Nir S Gov. Cell cluster migration: Connecting experiments with physical models. In *Seminars in Cell & Developmental Biology*. Elsevier, 2018.

[91] Peter Friedl, Joseph Locker, Erik Sahai, and Jeffrey E Segall. Classifying collective cancer cell invasion. *Nature Cell Biology*, 14(8):777–783, 2012.

[92] Amir Shamloo. Cell-cell interactions mediate cytoskeleton organization and collective endothelial cell chemotaxis. *Cytoskeleton*, 71(9):501–512, 2014.

[93] Adam Shellard, András Szabó, Xavier Trepat, and Roberto Mayor. Supracellular contraction at the rear of neural crest cell groups drives collective chemotaxis. *Science*, 362(6412):339–343, 2018.

[94] D Puiu and F Moldoveanu. Real-time path planner for arm robot collision avoidance. In *15th International Conference on System Theory, Control and Computing*, pages 1–6. IEEE, 2011.

[95] Johannes Nauta, Yara Khaluf, and Pieter Simoens. Hybrid foraging in patchy environments using spatial memory. *Journal of the Royal Society Interface*, 17(166):20200026, 2020.

[96] Surya G Nurzaman, Yoshio Matsumoto, Yutaka Nakamura, Kazumichi Shirai, Satoshi Koizumi, and Hiroshi Ishiguro. An adaptive switching behaviour between levy and brownian random search in a mobile robot based on biological fluctuation. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1927–1934. IEEE, 2010.

[97] Sabil Huda, Bettina Weigelin, Katarina Wolf, Konstantin V Tretiakov, Konstantin Polev, Gary Wilk, Masatomo Iwasa, Fateme S Emami, Jakub W Narojczyk, Michal Banaszak, et al. Lévy-like movement patterns of metastatic cancer cells revealed in microfabricated systems and implicated in vivo. *Nature Communications*, 9(1):1–11, 2018.