STOCHASTIC MAZE SOLVING UNDER THE GEOMETRIC AMOEBOT MODEL

By

AAYUSH YADAV

A thesis submitted to the

Graduate School—Camden

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Computer Science

written under the direction of

Dr. Sunil Shende

and approved by

Dr. Sunil Shende

Dr. Jean-Camille Birget

Dr. Desmond Lun

Camden, New Jersey

May, 2021

ABSTRACT OF THE THESIS

Stochastic Maze Solving Under the Geometric Amoebot Model

By AAYUSH YADAV

Thesis Director:

Dr. Sunil Shende

A self-organizing particle system (SOPS) is a collection of programmable units called *particles* with fixed computational capabilities. Within a particle system, individual particles execute fully distributed, local, asynchronous algorithms to achieve collective movement, configuration and co-ordination.

Leveraging recent developments in stochastic algorithm design for such systems, we propose a Markov chain based algorithm for collective maze-solving under the *geometric amoebot model* of SOPS for certain types of mazes. The inspiration for our algorithm comes from the algorithm for *phototaxing*, which uses external stimuli to create asymmetries in particle activity levels, thereby causing displacement of the particle system away from the stimuli.

Phototaxing has been proven for systems of two and three particles. We give an algorithm to verify phototaxing in arbitrarily large systems given the set of possible configurations are known. Using this algorithm, we verify that phototaxing also occurs in particle systems of size four.

Acknowledgements

This thesis is a culmination of the work of several people. I would like to thank Jesse Thomas, David Bushta, Christopher Till and Joshua Barnett for their help with the experimental setup.

Thank you, Dr. Sunil Shende, for supervising this thesis and the accompanying research. I am truly grateful for your support and guidance.

I also express my gratitude to other members of the thesis committee – Dr. Jean-Camille Birget and Dr. Desmond Lun, thank you for your valuable feedback on this work.

Most importantly, Erin, thank you for your constant support.

This work has partially been supported by NSF grant AF-1813940.

Dedication

For Erin. The amoebots did it!

Table of Contents

Chapter 1

Introduction

Systems of *programmable matter* find use in several areas including, but not limited to, swarm-robotics, bio-mimicry, synthetic biology and smart materials. A shared goal in this field is to realise computational (and eventually physical) models for artificial matter that can be applied to a variety of physical domains. One could envision a self-repairing paint material (see, for example, [12]) or a drug-delivery system involving safe transportation of pharmaceutical compounds as possible applications of programmable matter.

Abstracting away the details of area-specific application, we study the self-organization properties of such systems under the geometric amoebot model that was first theorised by Dolev et al. [17] as an "ameba-inspired [sic] system" of programmable particles and later introduced in an updated form by Derakhshandeh et al. [16]. The model is comprised of a system of particles, referred to as *amoebots*, that occupy the vertices of a triangular lattice G_Δ, and move along its edges. Individual amoebots execute distributed, local, asynchronous algorithms to achieve the broader goals of collective movement, configuration and co-ordination. Further, since amoebot decisions are made locally, the algorithms under this model can be viewed as emergent properties of the system and are performed with bounded memory related assumptions.

1.1 Contributions

Phototaxing [29] refers to the collective displacement of particles[1] away from (or towards) an external source of light. The algorithm for phototaxing forces asymmetries in particle activity levels via an external stimulus, thereby causing displacement of the particle

[1] we will often use the terms "particles" and "amoebots" interchangeably since all references to particles or particle-systems are made in the context of the amoebot model.

system. In this work, we study phototaxing in the amoebot model and propose an algorithm to verify whether phototaxing "provably occurs" in a system of amoebots. It has been shown that phototaxing "provably occurs", ie., occurs in expectation, for systems of two and three particles [29], but a general proof is not yet known. Our algorithm verifies expected phototaxing given that the set of all possible configurations up to rotation and translation are known in advance. This is a non-trivial requirement as the number of such configurations is exponential in the number of particles in the system [31], also referred to as its *size*. Using this algorithm we verify that phototaxing "provably occurs" in systems of size four.

We then consider the problem of maze-solving – i.e., the discovery of a path to the exit of a maze by a system of amoebots – and present a stochastic algorithm for collective maze-solving under the geometric amoebot model. Specifically, we introduce modifications in the amoebot design that allow for variable movement probabilities as a function of an individual amoebot's distance from the maze. This leads to a memory-less Markovian random-walk by the particle system that eventually leads to a path to the exit in specific types of mazes. This algorithm is similar to the algorithm for phototaxing, from which it is inspired, but with new parameters that allow the particles to adapt according to their position in the maze. We also analyse instances where this algorithm fails, leading to valuable insight into possible future work in this direction.

1.2 Related work

Systems of programmable matter can be classified as *passive* or *active*. Particles in passive systems lack autonomy, instead particle movement is determined by external factors such as environmental input and the system's structural configuration. Algorithms for tile based self-assembly [18, 27] and DNA computing [1, 3] are examples of passive systems. In contrast, particles in active systems possess the ability to make decisions about their actions, although it is possible that these decisions are partially guided by external factors. The amoebot model [17, 16, 10] and the *nubot* model [32], another model of self-assembly systems, both fall under the latter category of programmable matter systems.

Much of the early motivation behind the geometric amoebot model can be found in applications involving co-ordinated shape and pattern formation [15, 13, 14, 19], object coating [12, 6] and the formation of convex-hulls [8]. The model is given a detailed treatment by Daymude et al. [10]. More recently Daymude et al. [11] have also proposed a *deterministic* algorithm for distribution of energy within systems of amoebots. All of these algorithms are deterministic in nature and typically rely on careful state-management and protocols for inter-particle communication for their success.

The first *stochastic* algorithm under the amoebot model was given for compression and expansion of systems of amoebots [5]. The algorithm draws its inspiration from the Ising model of ferromagnetism [21] to define a Markov chain, on the set of valid system configurations, that favours approximately compressed (or expanded) configurations in the long run. The fact that state transitions can be restricted to correspond to local particle moves is indeed remarkable and has led to several further successes using this technique. Arroyo et al. [2] extended this methodology to given an algorithm for the formation of bridges over gaps in the triangular lattice. Similarly, Savoie et al. [29] demonstrated phototaxing in systems of amoebots. In the more recent work of Li et al. [23], the Markov chain based algorithms for problems of compression, expansion and collective transport were experimentally verified in a physical setup involving active "cohesive granular robots".

An abundance of instances of collective maze-solving exist in the natural world. Ants navigate unfamiliar and complex topologies by depositing a trail of pheromones that is then picked up by other ants in the colony and the process is repeated, eventually reaching a state of equilibrium where the entire colony follows the trail with the strongest scent [30]. Slime molds, often considered model organisms for studying biological self-organization, are known to optimally solve mazes by spreading their mass across the maze and then pruning any extensions that do not lead to an exit [26, 28]. Within the context of the amoebot model, the trivial "wall-follower" approach to maze-solving, wherein the system follows the nearest wall it can find until it finally exits the maze, is yet to be studied. A major shortcoming of such an algorithm is that in its most basic form, it fails to utilise the inherent parallelism offered by the amoebot model. The design of efficient deterministic algorithms for maze-solving under the amoebot model is of independent interest and in

this work, we restrict ourselves to designing a stochastic algorithm for maze-solving.

1.3 Organization

The remaining text is organized as follows. In Chapter 2 we introduce the geometric amoebot model including the algorithmic techniques used in the design of algorithms under this model. The stochastic algorithms under the model rely on defining a Markov chain so an exposition to Markov chains and mixing times is also given. Chapter 3 elaborates on the algorithms for compression [5] and phototaxing [29] as they form the basis of our results. We then give the phototaxing verification algorithm in Chapter 4, the algorithm for collective maze-solving in Chapter 5 and finally end with our concluding remarks in chapter 6.

1.4 Notation

Throughout the text, we refer to an individual programmable particle by the notation $\mathcal{P}$. This is different from any allusion to $\mathbf{P}$, the matrix of transition probabilities (see Section 2.4). Particularly, we reserve boldface lettering for matrices (uppercase) and vectors/vector valued functions (lowercase). When necessary, individual elements of matrices and vectors are denoted by their corresponding plain, lowercase lettering with appropriate indices. For instance, consider the scalar valued function $g : \mathcal{X} \to \mathcal{Y}$ for some sets $\mathcal{X}$ and $\mathcal{Y}$. We then define $\boldsymbol{g}$ as the vector corresponding to $g(x)$ applied to every $x \in \mathcal{X}$. Similarly, elements of the matrix $\mathbf{P}$ are denoted by $p_{i,j}$ where i and j are the row and column indices respectively. The lowercase letter n is always used in reference to the size of a system of programmable particles. Equivalently, it is the number of particles in a given system.

Chapter 2

Background and the Model

In this chapter we expand on several standard definitions and results under the Geometric Amoebot model. We also discuss deterministic and stochastic approaches to algorithm design that have been studied under this model. For an in-depth survey of the topics in this section, we direct the reader to the book chapter by Daymude et al. [10].

2.1 The Amoebot Model

The geometric amobeot model is an abstract model of self-organizing particle systems. It consists of individual computational elements (or particles) known as *amoebots* represented as unique points on an infinite triangular lattice G_Δ. An amoebot may exist in a *contracted* state (see figure 2.1) or in an *extended* state wherein the particle occupies two adjacent locations in the lattice. Particles move via a series of alternating *extensions* and *contraction* steps along the edges, as demonstrated in Figure 2.1 (b). This movement is reminiscent of the amoeba after which the particles in this model have been named.

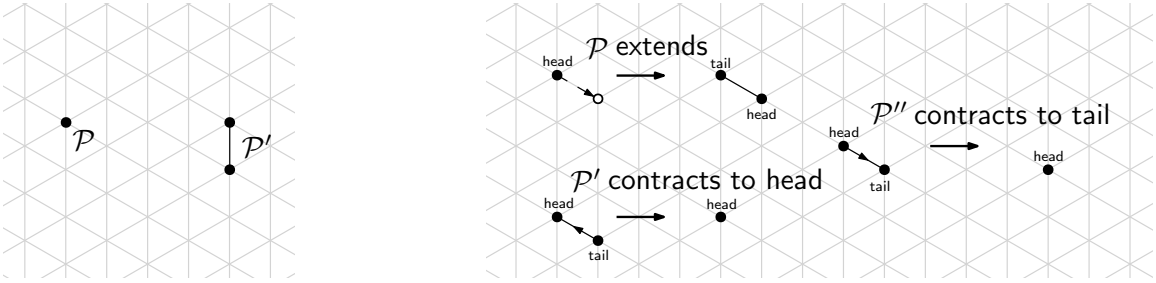

Figure 2.1: (a) A contracted amoebot $\mathcal{P}$ and an expanded amoebot $\mathcal{P}'$; and (b) Valid particle moves for $\mathcal{P}$ in a contracted state, and for $\mathcal{P}'$ and $\mathcal{P}''$ in extended states.

All amoebots must keep track of their *port labels* that uniquely identify the edges surrounding it. Further, the particles are *anonymous* in that they lack any global identifier, however they can locally identify their *neighbours* by the port labels corresponding to the connecting edge. It is also assumed that there is no shared co-ordinate system or a global sense of orientation between particles.

Each particle has a constant amount of local memory that stores its extended/contracted status, the port labellings and any other state related information directly relevant to the application at hand. In addition, particles can also communicate with their immediate neighbours on the account of having read and write access to their local memory stores.

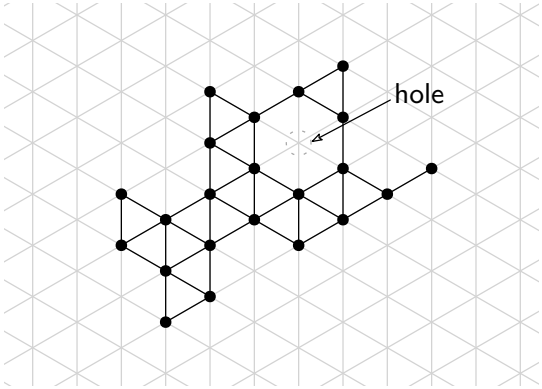

Figure 2.2: A sample configuration G with a *hole*.

Computations in the model are performed as *atomic actions* realised by the $\mathcal{A}$sync model of distributed computing. An atomic action for the amoebot model is the *activation* of one amoebot. A standard result under this model is that for any concurrent asynchronous execution of atomic actions there exists (subject to conflict resolution) a sequential ordering of actions that returns the same output. This greatly simplifies analysis of distributed algorithms under the amoebot model. Additionally, an *asynchronous round*, or simply a round, is completed when every particle has activated at least once. This assumes that all activations are fair, i.e., at any given time, all particles have an equal probability of activation. In this work, we only consider the end states of the atomic action where particles have either completed a single move or stayed-put in their position before activation, thus always remaining contracted. This allows us to define a *configuration* of a system of

amoebots as a sub-graph $G = (V, E)$ of the lattice, where V is the set of all positions occupied by the amoebots relative to the system on G_Δ and E is the collection of edges between two adjacent amoebots in G_Δ.

The sub-graph G is also typically constrained to remain *connected*. The justification for this is that because a disconnected system is unlikely to reconnect via strict local moves, and because particles can only communicate with other adjacent particles, communication is severely hindered in such a system. An exception to this constraint is the algorithm proposed by Li et al. [23] where this constraint was recently shown to be superfluous for the purpose of aggregation of systems of amoebots. All algorithms we discuss assume the connectivity constraint. Algorithms derived from the algorithm compression [5] also require the introduction of the concept of *holes*. A hole in a configuration is a maximal connected component of $G_\Delta \setminus G$ that is also finite.

A note on random number generation. The limited memory requirement of the model also limits the number of random bits held by each particle. A particle may therefore only hold random values of constant precision, although the exact number of bits may be determined based on the application.

2.2 Deterministic algorithms under the amoebot model

The geometric amoebot model was invented, in its most current form, as a theoretical model for programmable matter with self-organizing capabilities [16]. The need for such a model was primarily motivated by real world applications in shape and pattern formation [15, 13, 14] and the coating of objects [12, 6] by systems of programmable matter.

Deterministic algorithms under the geometric amoebot model comprise a class of carefully designed distributed, local, asynchronous protocols executed by individual amoebots to (deterministically) self-organize into a target configuration. These algorithms depend on such algorithmic primitives as *leader election* [15, 9] and the *spanning forest primitive* [15], for their operational requirements.

To give a high level idea of how these primitives utilise the full capabilities of the amoebot model, we briefly describe the "improved leader election" algorithm by Daymude et al. [9]. In leader election, a system of particles elects one particle that declares itself as the leader and no other particle declares itself the leader. Leader particles are often necessary to set off the execution of deterministic algorithms, to initialise particle states and ensure certain application specific conditions are met. To solve leader election, Daymude et al. [9] propose an asynchronous token passing scheme to create chains of contiguous particles on the boundary (including those around holes) of a particle system configuration. Progressing through several phases of communication, a particle on the unique outer-boundary that unequivocally leads the longest such chain declares itself the leader. Since the chains are initialised by random coin flips, this algorithm provably solves the problem of leader election *with high probability* (w.h.p.)[1] [9].

In the problem of basic shape formation, a system of amoebots can re-configure, in number of rounds linear in the size of the system, to form a line, a triangle or a hexagon on the triangular lattice [15, 13]. A more general version of shape formation [14, 19] allows for re-configuration of the particle system into a broader class of shapes that can be built by attaching triangles to some starting shape in $\mathcal{O}(\sqrt{n})$ number of rounds w.h.p. The bound is probabilistic due to its dependence on the leader election primitive. However, once a leader has been elected, the algorithm for general shape formation is fully deterministic.

The problem of object coating involves covering an object (or a part of an object) with a layer of programmable matter. An immediate application of this can be seen in the design of self-repairing paint or construction materials. The universal coating algorithm [12, 6] correctly solves the object coating problem in $\mathcal{O}(n)$ number of rounds in the worst case, w.h.p. Where once again the randomness comes from the leader election primitive. Similar to object coating is the problem of wrapping of an object by a convex hull, this is solved by Daymude et al. [8] in number of rounds linear in the length of the object's boundary.

[1] specifically, with probability $1 - 1/n^{\mathcal{O}(1)}$

2.3 Stochastic algorithms under the amoebot model

Stochastic algorithms under the geometric amoebot model rely on individual particles making probabilistic decisions in order to reach some near-optimal state with respect to a pre-determined objective. The approach here is markedly different from deterministic algorithms discussed in the previous section in that, here, particles carry out minimal communication and no state management whatsoever. Instead, the standard technique is to design a Markov chain, defined over the particle-system configuration, whose long run probabilities favour certain desirable configurations. The notion of a desirable configuration is captured by an *energy function* that quantifies some measurable aspect of the system such as the number of edges, triangles, particles on the perimeter of the system etc. The Markov chain must be designed keeping in mind that the resulting algorithm should transfer easily in a distributed, local, asynchronous paradigm executed independently on each particle.

Markov chains are used extensively in statistical physics to sample configurations of physical systems with the probability of any configuration, σ – at temperature τ, and whose energy is determined by a *Hamiltonian* $H(\sigma)$ – given by the quantity $w(\sigma)/Z$ where $w(\sigma) = e^{-H(\sigma)/\tau}$ is the *weight* of the configuration and $Z = \sum_{\sigma'} w(\sigma')$ is a normalising constant known as the *partition function*. The same physical inspiration is extended to design Markov chains for stochastic algorithms under the amoebot model. The Markov chain based technique has had remarkable success in several recent objectives including algorithms for compression and expasion [5], phototaxing [29], shortcut-bridging [2], and seperation [4] in systems of programmable particles.

2.4 Markov chains and mixing times

Markov chains have been the tool of choice in the design of stochastic algorithms under the geometric amoebot model [5, 29, 2, 4, 23]. In this section, we review the theory of Markov chains and their mixing times; the analysis of mixing time, or the time to convergence of Markov chains turns out to be essential in proving that the system does indeed reach the desired state in the long run. The reader may also refer to the text by Levin, Peres and

Wilmer [22] for a more complete exposition of Markov chains and their mixing times.

Let Ω be the state space consisting of all particle system configurations. A *Markov chain*, $\mathcal{M}$, on Ω is a memory-less random process, i.e., the transitions from one state to another are made independently of time, with the probability of transition depending only on the current state of the chain. Assuming Ω to be finite and discrete, we may completely specify $\mathcal{M}$ by the use of a *transition matrix*, $\mathbf{P} \in [0,1]^{|\Omega|\times|\Omega|}$ defined such that $p_{\omega,\nu}$ is the probability of transitioning from state ω to state ν in a single step for any $\omega, \nu \in \Omega$. Further, the quantity, $\mathbf{P}^t[\omega \to \nu]$ gives the t-step transition probability of reaching state ν from a state ω in exactly t transitions of the Markov chain, where we once again have $\omega, \nu \in \Omega$.

Ergodicity. A Markov chain is *irreducible* if there is a time-step t_0 such that for all $t > t_0$, $\mathbf{P}^t[\omega \to \nu] > 0$ for all $\omega, \nu \in \Omega$ or, in words: it is always possible to get from one state to any other state through a sequence of transitions. A Markov chain is *periodic* if its states can be grouped into $n > 1$ disjoint subsets $S_0, S_1, \ldots, S_{n-1}$ such that all states in S_i transition into S_{i+1} (modulo n) for $i = 1, 2, \ldots, n-1$. Then, we may say that the Markov chain is *aperiodic* if such a grouping of states is not possible. More formally, we may say that a Markov chain is *aperiodic* if for all $\omega \in \Omega$, $\gcd\left\{t : \mathbf{P}^t[\omega \to \omega] > 0\right\} = 1$. Finally, a Markov chain that is both irreducible and aperiodic is *ergodic*.

Stationary distribution of $\mathcal{M}$. A probability distribution π over Ω is the *stationary distribution* of a Markov chain if $\pi\mathbf{P} = \pi$. The time taken to reach this stationary distribution is known as the *mixing time* of the Markov chain. Ergodic Markov chains over a finite state space have the nice property that they converge to a unique stationary distribution that, for any $\omega, \nu \in \Omega$, is given by $\pi_\nu = \lim_{t\to\infty} \mathbf{P}^t[\omega \to \nu]$. The uniqueness of the stationary distribution can be explained intuitively by looking closely at the requirements for ergodicity presented above – irreducibility prevents "absorbing states" or states that can not be escaped, and without aperiodicity stationarity can not be reached.

It is worth noting that finding the mixing time for the Markov chain in the context of the amoebot model is a non-trivial open problem; it is, however, possible to check that a distribution π' is the unique stationary distribution by verifying the, stronger, *detailed*

balance condition $\pi'_\nu p_{\nu,\omega} = \pi'_\omega p_{\omega,\nu}$. In fact, we use detailed balance in Section 3.1, to show that the Markov chain for the compression algorithm converges to a unique stationary distribution.

Metropolis filter. We now give a general description of the Markov chain for compression as it broadly applies to all stochastic algorithms under the model while deferring the task of defining compression in the amoebot model to the next chapter. To that end, consider a Markov chain $\mathcal{MC}_{\text{compress}}$, with stationary distribution π, over the state space Ω, comprised of all the possible configurations in a connected particle system of fixed size. For any $\omega, \nu \in \Omega$, state-transitions of the form $\omega \to \nu$ correspond to a single particle move that occurs with probability $p_{\omega,\nu} = \min\left\{1, \frac{\pi_\nu}{\pi_\omega}\right\}$ (and $p_{\omega,\omega}$ can be found by complementation). We say that ν is in the *neighbourhood* of ω if $p_{\omega,\nu} > 0$. This method of setting transition probablities is known as the Metropolis-Hastings [20] algorithm, that eventually leads a sampling of states according to the stationary distribution[2]. It is not immediately obvious that these transition probabilities can be computed locally by every particle. However, as we describe in Section 3.1.1, that does turn out to be the case.

2.5 Physical inspiration and the Ising model

In this section, we explain how the stochastic algorithms for the amoebot model derive their inspiration from the Ising model [21] of statistical physics. This parallelism will help us see the amoebot model in a new light: as a variant of the Ising model in the 2-dimensions with fixed boundary conditions.

The design of the Markov chain for stochastic algorithms under the Amoebot model is motivated by statistical physics with, perhaps, the most direct analogue being the widely studied Ising model for ferromagnetism [21]. Conceptually, the model consists of a physical system represented by a lattice with positive or negative (alternatively up or down)

[2]Historical note: some controversy exists concerning the name of the algorithm; authors Rosenbluth and Rosenbluth of the original paper [25] maintain that the algorithm is their invention and that Metropolis simply provided computing time.

spins assigned to each vertex and an associated temperature. More formally, we consider a regular lattice graph where $[n] = \{0, 1, \ldots, n-1\}$ are the vertices that we call *sites*; and E is the collection of pairs of sites $(i, j) \in [n]$ with non-zero *interaction energy*, V_{ij}. A configuration, $\sigma = \{\sigma_i\}_{i \in [n]}$ is, then, an assignment of positive/up ($\sigma_i = +1$) and negative/down ($\sigma_i = -1$) spins to all the sites on the graph, and the *energy* of a configuration under an external field B is given by its Hamiltonian $H(\sigma) = -\sum_{\{i,j\} \in \mathrm{E}} V_{ij}\sigma_i\sigma_j - B\sum_{k \in [n]} \sigma_k$. Observe here that the first term in the Hamiltonian corresponds to the energy due to inter-site interaction and the second term to the energy due to the external field. The probability that a system is in a state (or configuration) σ at equilibrium is $\exp(-\beta H(\sigma))/Z$ where $\beta > 0$ is inversely proportional to the temperature and $Z = Z(V_{ij}, B, \beta) = \sum_{\sigma \in \{-1,+1\}^n} \exp(-\beta H(\sigma))$ is the partition function.

In the amoebot model, we consider nodes in the triangular lattice G_Δ to have a positive spin value if it is occupied, and negative otherwise. The amoebot model is then similar to an Ising model with *fixed magnetization*, wherein the total number of spins of each kind does not change. This naturally follows from the fact that the size of the system remains fixed. We also require the additional constraint that the system of amoebots is simply-connected, a requirement that is non-standard in the Ising model but necessary for the proofs of Markov chain convergence in the compression algorithm [5]. For a system without the simple-connectivity constraint, a mapping to the fixed-magnetization Ising model is known [23].

Chapter 3

Stochastic Algorithms

Compression and Phototaxing

Stochastic algorithms under the geometric amoebot model are designed by specifying a Markov chain over the set of system configurations and then sampling the states based on the desired objective. This technique was first given by Cannon et al. [5] for an algorithm for *α-compression* of SOPS, and has since resulted in stochastic algorithms for several other objectives under the amoebot model [29, 2, 4, 23].

We now describe the algorithms for compression and phototaxing, as the former is required for understanding the stochastic approach and the latter motivates both of our main results. At the end the upcoming section we also comment on the challenges in proving reasonable bounds on the mixing times for the Markov chain defined under the compression scheme, and by extension, under the existing class of stochastic algorithms for the amoebot model.

3.1 Compression

The problem of compression in a system of self-organizing particles is quite simply the aggregation of all the particles into a ball of the smallest radius in an appropriately defined space. Aggregation is considered typical swarm behaviour, swarms of insect species such as ants, cockroaches and mosquitoes are known to aggregate for the purpose of foraging, reproduction etc. It is remarkable that these co-operative agents are able to accomplish group level co-ordination through individual action. Compression in general SOPS is a difficult problem for several reasons, not least because of the complexity of the underlying inter-particle communication protocols thus involved.

We now express compression in the context of the geometric amoebot model. Let $p(\sigma)$ denote the perimeter of a system configuration σ, then $p(\sigma)$ is the sum of the lengths of all boundaries (including those surrounding holes) of σ. Define $p_{\min} \equiv \min_{\{\sigma \in \Omega : |\sigma| = n\}} p(\sigma)$ as the minimum possible perimeter among all systems of size n. A simply-connected configuration σ is said to be α-compressed if $p(\sigma) \leq \alpha \cdot p_{\min}$ for any $\alpha > 1$. By simply-connected, we mean that the configuration σ is connected and it does not contain any holes.

3.1.1 A Markov chain algorithm for compression

The Hamiltonian of a compressing system in configuration σ is defined to be proportional to the perimeter, $p(\sigma)$. As a result, a configuration with smaller perimeter also has lower overall energy. However, while the goal of compression is to minimize the perimeter (or equivalently the radius), the following lemma due to Cannon et al. [5] suggests a more convenient choice of measure

Lemma 3.1.1 (Cannon et al. [5]) *A simply-connected particle system configuration with minimum perimeter is also a configuration with the maximum number of edges and the maximum number of triangles.*

In Section 2.5 we noted that the Metropolis probabilities can be calculated locally. It is through Lemma 3.1.1 that this is made possible. Specifically, it allows us to write the Hamiltonian as $H(\sigma) = -\varepsilon(\sigma)$, where $\varepsilon(\sigma)$ is the number of edges in a configuration σ and the weight of the configuration is then $w(\sigma) = e^{-H(\sigma)/\tau} = \lambda^{\varepsilon(\sigma)}$ where the parameter λ controls the relative strength of compression ($\lambda > 1$ favours configurations with more edges).

To correctly analyse the Markov chain, certain necessary constraints must be maintained during the execution of the algorithm for compression. The main *structural invariants* are that (i) connectivity should be maintained at all times; and (ii) particle moves should not result in new holes. Once simple-connectivity has been reached, the fact that it is maintained is evident by construction. To maintain these structural invariants, any activated particle must satisfy one of two properties in moving from some vertex location ℓ to an adjacent ℓ' on G_Δ. Before specifying the properties, some notation must be established.

For a particle $\mathcal{P}$ at location ℓ, let $N(\ell)$ denote the set of particles in the neighbourhood of ℓ. For adjacent locations ℓ and ℓ', let $\mathbb{U}$ and $\mathbb{S}$ denote the sets $\{N(\ell) \cup N(\ell')\} \setminus \{\ell, \ell'\}$ and $N(\ell) \cap N(\ell')$ respectively. Note that $|\mathbb{S}|$ can be one of 0, 1 or 2. The properties are then stated as follows:

> Property 1. $|\mathbb{S}| \in \{1, 2\}$ *and every particle in* $\mathbb{U}$ *is connected to a particle in* $\mathbb{S}$ *by a path through* $\mathbb{U}$.

> Property 2. $|\mathbb{S}| = 0$, ℓ *and* ℓ' *each have at least one neighbour, all particles in* $N(\ell) \setminus \{\ell'\}$ *are connected by paths within this set, and all particles in* $N(\ell') \setminus \{\ell\}$ *are connected by paths within this set.*

Intuitively, Property 1 is preventing certain new holes, Property 2 is preventing the system from disconnecting, and together these conditions ensure eventual ergodicity of the resulting Markov chain. With this background, we now describe the compression algorithm as given by Cannon et al. [5].

Assuming a uniform probability of activation across all particles in a configuration, the Markov chain $\mathcal{MC}_{\text{compress}}$ can now be formally defined as in Algorithm 1 for a single particle $\mathcal{P}$.

Algorithm 1 (Cannon et al. [5]) A Markov chain compression algorithm for particle $\mathcal{P}$.

Given: the compression parameter λ.

1: Particle $\mathcal{P}$, at location ℓ of G_Δ, chooses a location ℓ' uniformly at random from all its six neighbouring locations.
2: **if** (i) ℓ' is unoccupied, (ii) Property 1 or Property 2 hold, and (iii) a move from ℓ to ℓ' does not *create* a hole **then**
3: $\quad$ $\mathcal{P}$ generates a random number $q \in (0, 1)$.
4: $\quad$ Let ε be the number of edges of a configuration incident on $\mathcal{P}$ at position ℓ, and similarly ε' the number of edges incident on $\mathcal{P}$ if it were at ℓ'.
5: $\quad$ **if** $q < \lambda^{\varepsilon' - \varepsilon}$ **then**
6: $\quad\quad$ $\mathcal{P}$ moves to ℓ'.
7: $\quad$ **else**
8: $\quad\quad$ $\mathcal{P}$ stays at ℓ.

Observe here that locally checking if a number q drawn uniformly from $(0, 1)$ is less than $\lambda^{\varepsilon' - \varepsilon}$ is exactly the *Metropolis filter* we discussed in Section 2.4. We reproduce $\mathcal{MC}_{\text{compress}}$

(see figure 3.1) and confirm that for sufficiently large λ, the Markov chain leads to α-compression of any initially connected configuration. We demonstrate a sample result from our simulations in Figure 3.1 using only a constant number of bits for q.

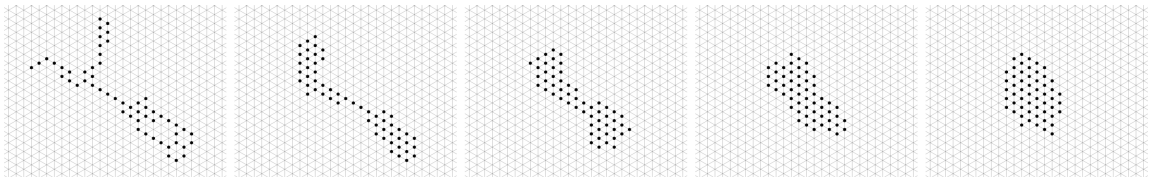

Figure 3.1: A simulation of the compression algorithm for 55 particles over approximately 3 million iterations.

A distributed, local, asynchronous algorithm for compression . A central feature of the amoebot model is that individual particles are able to distribute computation and execute local algorithms asynchronously. Transforming $\mathcal{MC}_{\text{compress}}$ into a distributed, local, asynchronous algorithm, $\mathcal{A}_{\text{compress}}$, for compression requires two major changes, namely, introducing asynchronous particle activations, and decoupling particle extension and contraction during a move into separate activations. The latter being necessary because the amoebot model only permits one move per activation.

Asynchronous activation of particles, by itself, does not suggest uniform probability of activation across all particles; a property that is useful in the analysis of $\mathcal{MC}_{\text{compress}}$, explicitly allowing us to compute its stationary distribution. To overcome this, each particle is given its own *Poisson clock* with mean 1. After having completed an activation, a particle will next activate after a *time-delay*, t, drawn with probability e^{-t}, i.e., the Poisson distribution with mean 1. This guarantees that, at any given time, activation probabilities are drawn from the same exponential distribution, and all updates can be performed locally. In addition, each particle also stores a local FLAG variable that stores the extension/contraction status of its neighbourhood.

By enforcing stronger conditions on particle movement, the asynchronous algorithm $\mathcal{A}_{\text{compress}}$ is shown to correctly emulate $\mathcal{MC}_{\text{compress}}$. All analysis in the next section is restricted to the Markov chain algorithm $\mathcal{MC}_{\text{compress}}$, and the extension to Algorithm $\mathcal{A}_{\text{compress}}$ follows. The main theorem of α-compression is now given.

Algorithm 2 (Cannon et al. [5]) A distributed, local, asynchronous compression algorithm for particle $\mathcal{P}$.

Given: the compression parameter λ.

When $\mathcal{P}$ is contracted:

1: Particle $\mathcal{P}$, at location ℓ of G_Δ, chooses a location ℓ' uniformly at random from all its six neighbouring locations.
2: **if** (i) ℓ' is unoccupied and (ii) no other particles in $\mathcal{P}$'s neighbourhood are extended **then**
3: $\mathcal{P}$ extends to occupy ℓ and ℓ'.
4: **if** the neigbourhood $\mathbb{U}$ does not have any extended particles **then**
5: $\mathcal{P}$ sets its FLAG to TRUE.
6: **else**
7: $\mathcal{P}$ unset its FLAG to FALSE.

When $\mathcal{P}$ is extended:

8: **if** (i) ℓ' is unoccupied, (ii) PROPERTY 1 or PROPERTY 2 hold, (iii) a move from ℓ to ℓ' does not *create* a hole, and (iv) FLAG is set **then**
9: $\mathcal{P}$ generates a random number $q \in (0,1)$.
10: Let ε be the edges of a configuration incident on $\mathcal{P}$ at position ℓ, and similarly ε' the edges incident on $\mathcal{P}$ if it were at ℓ'.
11: **if** $q < \lambda^{\varepsilon'-\varepsilon}$ **then**
12: $\mathcal{P}$ contracts to ℓ'.
13: **else**
14: $\mathcal{P}$ contracts back to ℓ.

Theorem 3.1.1 (Cannon et al. [5]) *For any $\lambda > 2+\sqrt{2}$, and for any constant $\alpha > \log_{2+\sqrt{2}} \lambda/(\log_{2+\sqrt{2}} \lambda - 1)$ there exists and $n^* \geq 0$ and $\zeta < 1$ such that for all $n \geq n^*$, a random sample σ drawn according to the stationary distribution π of $\mathcal{MC}_{compress}$ satisfies*

$$\mathbb{P}_{\sigma\sim\pi}[p(\sigma) \geq \alpha \cdot p_{min}] < \zeta^{\sqrt{n}}.$$

Proof sketch.

The first step is to show that the Markov chain is reversible. This can be proved by first observing that since all state transitions correspond to a single particle move, a move is reversed when the particle moves back to its initial position. Now, since particles only move to adjacent locations and both PROPERTY 1 and PROPERTY 2 are symmetric for adjacent locations, with positive probability a move can be reversed.

Cannon et al. [5] also show that any configuration can be turned into a straight-line

configuration in a sequence of valid steps. Then, by reversibility, the straight-line configuration can be turned into any other hole free configuration in Ω^*. Clearly, $\mathcal{MC}_{\text{compress}}$ is irreducible. Further, by design, there is positive probability that an activated particle does not move – also known as *self-loop* transition – and so $\mathcal{MC}_{\text{compress}}$ is also aperiodic. Therefore, it is ergodic on Ω^*. Now, since the Markov chain is ergodic and finite, it must be unique.

Next, it must be shown that the Gibbs distribution according to $w(\sigma)$ is indeed the unique stationary distribution of the Markov chain $\mathcal{MC}_{\text{compress}}$. This can be proved by verifying the detailed balance conditions and individually considering cases where, in one step of the Markov chain, the number of edges either decreases or strictly increases.

The remaining part of the proof hinges on establishing a sufficiently tight upper bound on $\sum_\sigma \pi_\sigma$, the probability that the Markov chain is in some configuration σ whose perimeter is at least $\alpha \cdot p_{\min}$ at stationarity. This requires analysing the geometry of the triangular lattice and thus appropriately bounding the number of such configurations.

A short note on mixing time of $\mathcal{MC}_{\text{compress}}$

The analysis of mixing time of $\mathcal{MC}_{\text{compress}}$ uses much of the same machinery as the Ising model given their similarities. In fact, for the Ising model with local update dymanics and fixed-boundary coundition, the best known upper bound is quasipolynomial [24]. Similar challenges in proving tighter upper bounds seem to appear for the Markov chain for compression [5]. While rigorous bounds for convergence are not known, Cannon et al. [5] have speculated that compression may happen well before the Markov chain has fully mixed. With that, Cannon et al. [5] conjecture that α-compression occurs between $\Omega(n^3)$ and $\mathcal{O}(n^4)$ rounds of $\mathcal{MC}_{\text{compress}}$, and equivalently between $\Omega(n^2)$ and $\mathcal{O}(n^3)$ asynchronous rounds of $\mathcal{A}_{\text{compress}}$.

3.2 Phototaxing

The probability of movement in Algorithm $\mathcal{MC}_{\text{compress}}$ is uniform across all legal directions for all particles. By disturbing this symmetry, one might expect to induce displacement of the system in a desired direction. Savoie et al. [29] demonstrate that this is indeed the case.

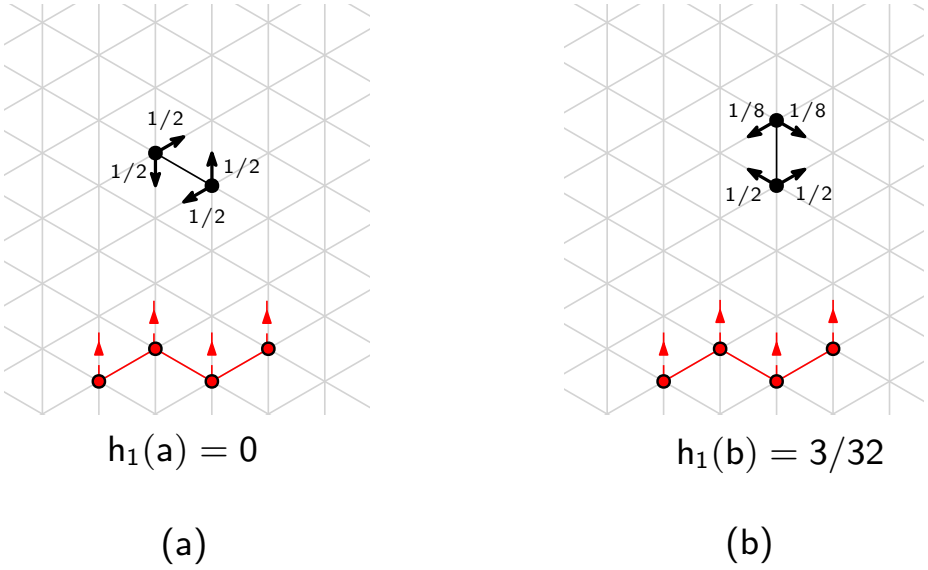

Figure 3.2: Two possible configurations, unique up to rotation and translation, in a system of size two. The red line marks the source of the signal and the arrows indicate direction. The probability of movement, obtained according to Algorithm 3, in the marked directions (bold black lines) are indicated for each particle assuming that it activates next.

In the phototaxing set-up, all edges of the lattice graph G_Δ are assumed to be of unit length; and a fixed, continuous line of point sources on the vertices of G_Δ broadcasts rays along the lattice line, in a fixed vertical direction toward a connected particle system. A particle is *unoccluded* from light if it is the first particle to receive the ray on its vertical lattice line. A particle that is not unoccluded is *occluded* from light by another particle. Particles are sensitive to the signal and can locally determine whether they are in an occluded or unoccluded state upon activation. Further, the y co-ordinate of the center of mass of a particle system starting in some initial configuration, σ_0, is referred to as its *height*, y_0, with respect to the fixed jagged line of point sources itself at height 0 or $-1/2$. It is important to note that all height related calculations are done with respect to this fixed

reference line.

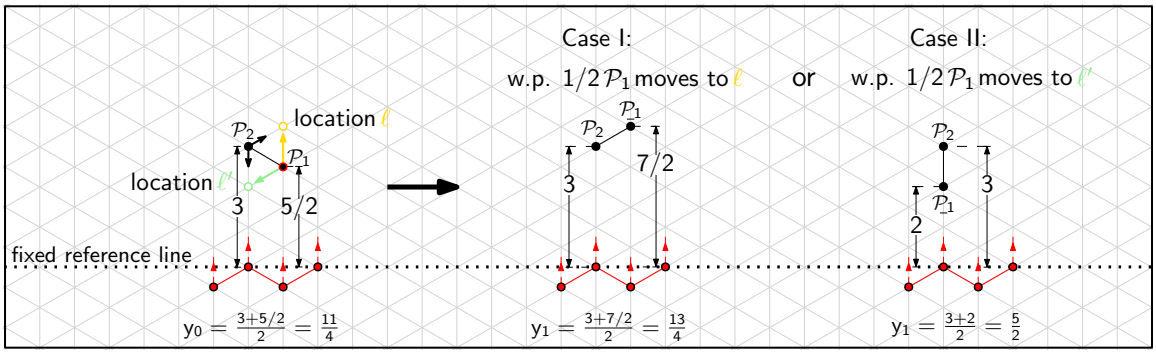

Figure 3.3: One step transition of particle $\mathcal{P}_1$ to ℓ or ℓ' and the corresponding heights of the resulting system with respect to the fixed reference line.

Figure 3.3 illustrates the phototaxing setup for the two-particle case. An activating $\mathcal{P}_1$, in some configuration at starting height y_0, moves to one of the two locations ℓ and ℓ' resulting in the height of the system respectively increasing and decreasing to y_1.

With this background, the phototaxing algorithm is quite simply the execution of the subroutine described in Algorithm 3 by each activated particle. For a system of programmable particles, phototaxing is said to *provably occur* if after a finite number of activations, the height of the particle system has strictly increased in expectation. This is a slight simplification of termi nology by the original authors as phototaxing only occurs

Algorithm 3 (Savoie et al. [29]) Phototaxing subroutine of each particle $\mathcal{P}$.

1: **if** $\mathcal{P}$ is *unoccluded* **then**
2: Execute Algorithm $\mathcal{MC}_{\text{compress}}$
3: **else**
4: Execute Algorithm $\mathcal{MC}_{\text{compress}}$ with probability 1/4.

in expectation and from this point on we state is as such. For small systems, we have the following theorem due to Savoie et al.

Theorem 3.2.1 (Savoie et al. [29]) *For systems of two and three particles, phototaxing occurs in expectation.*

In figures 3.2 and 3.4 we see the set of possible states unique up to translation and reflection for two- and three-particle systems respectively, along with the probabilities

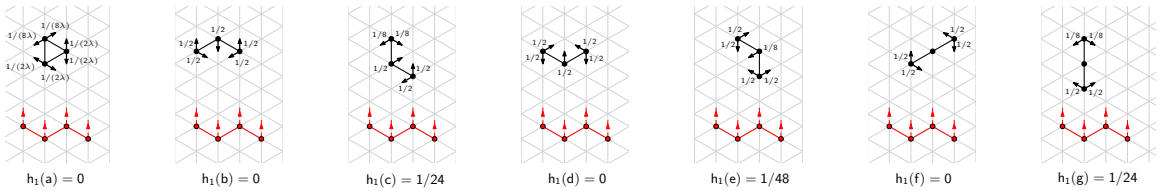

Figure 3.4: Seven possible configurations in a system of size three and the corresponding one-step expected change in height values.

of individual particle movement that are obtained according to Algorithm 3. It should be noted that the quantity λ in the transition probabilities is the same compression parameter as in the previous section. We now show how to express the expected change in height algebraically.

For any $\omega, \nu \in \Omega$, let $\Pr(\omega \to \nu)$ be the probability of reaching configuration ν from ω in one step. Also, starting in some configuration σ_0 at height y_0, let $\sigma_1, \sigma_2, \ldots$ be the sequence of state transitions for $\sigma_0, \sigma_1, \ldots$ in the state space, and let $y_1, y_2, \ldots$ be the corresponding height according to each transition. For $t \in \mathbb{N}$, define $h_t : \Omega \to \mathbb{Q}^+$ as the t-step expected change in height for a given starting configuration. Accordingly, the one-step expected change in height is

$$h_1(\sigma_0) \equiv \mathbb{E}[\Delta_1 y \mid \sigma_0] = \sum_{\sigma_1 \in \Omega} (y_1 - y_0) \Pr(\sigma_0 \to \sigma_1) \tag{3.1}$$

where $\Delta_1 y$ represents the change in the height going from σ_0 to any σ_1 in one step. In other words, $h_1(\sigma_0)$ is the expected change in height in one state transition, starting in a configuration σ_0. Likewise, in two steps

$$
\begin{aligned}
h_2(\sigma_0) \equiv \mathbb{E}[\Delta_2 y \mid \sigma_0] &= \mathbb{E}[y_2 - y_0 \mid \sigma_0] = \mathbb{E}[y_2 - y_1 + y_1 - y_0 \mid \sigma_0] \\
&= \sum_{\sigma_1} \mathbb{E}[(y_2 - y_1) + (y_1 - y_0) \mid \sigma_0, \sigma_1] \Pr(\sigma_0 \to \sigma_1) \\
&= \sum_{\sigma_1} (\mathbb{E}[(y_2 - y_1) \mid \sigma_0, \sigma_1] + \mathbb{E}[(y_1 - y_0) \mid \sigma_0, \sigma_1]) \Pr(\sigma_0 \to \sigma_1) \\
&= \sum_{\sigma_1} (\mathbb{E}[\Delta_1 y \mid \sigma_1] + (y_1 - y_0)) \Pr(\sigma_0 \to \sigma_1) \\
&= \sum_{\sigma_1} h_1(\sigma_1) \Pr(\sigma_0 \to \sigma_1) + \sum_{\sigma_1} (y_1 - y_0) \Pr(\sigma_0 \to \sigma_1) \\
&= \sum_{\sigma_1} h_1(\sigma_1) \Pr(\sigma_0 \to \sigma_1) + h_1(\sigma_0)
\end{aligned}
$$

For the purpose of illustration, we also note the expression in three steps.

$$
h_3(\sigma_0) \equiv \mathbb{E}[\Delta_3 y \mid \sigma_0] = \sum_{\sigma_1} \underbrace{\left(\sum_{\sigma_2} h_1(\sigma_2) \Pr(\sigma_1 \to \sigma_2) + h_1(\sigma_1) \right)}_{h_2(\sigma_1)} \Pr(\sigma_0 \to \sigma_1) + h_1(\sigma_0)
$$

That is, in the general case of $t \geq 2$ state transitions starting in σ_0 can be written as the following recurrence

$$
h_t(\sigma_0) \equiv \mathbb{E}[\Delta_n y \mid \sigma_0] = \sum_{\sigma_1} h_{t-1}(\sigma_1) \Pr(\sigma_0 \to \sigma_1) + h_1(\sigma_0) \tag{3.2}
$$

The proof of Theorem 3.2.1 involves separately calculating the expected change in height in an iterative manner until, for some positive integer N, $h_N(\cdot) > 0$ for every individual configuration. Note that this is only possible in the case of two and three particles where the expected change in height in one step is either positive or zero for all configurations as seen in figures 3.2 and 3.4, and is not generally true for systems of larger sizes. It should also be mentioned that the Markov chain for Algorithm 3 is exactly the Markov chain for compression and the same results concerning convergence and mixing times apply here as well.

Chapter 4

Phototaxing Verification

Experimental evidence suggests the phototaxing occurs in systems of arbitrary sizes [29]. With the goal of finding a general proof for phototaxing, we study the amoebot system of four particles. In order to do that, we first generalise the proof technique of Savoie et al. [29] and then give an algorithm for verifying that phototaxing occurs in expectation. Using this algorithm, we extend Theorem 3.2.1 to systems of four particles.

4.1 The algorithm for verification

Let $\mathbf{P} \in [0,1]^{|\Omega|\times|\Omega|}$ denote the transition matrix with entries p_{ij} for $i,j \in \{1,2,\ldots,|\Omega|\}$ such that for all $\omega_i, \omega_j \in \Omega$, $p_{ij} = \Pr(\omega_i \to \omega_j)$, and $\mathbf{Y} \in \mathbb{Q}^{|\Omega|\times|\Omega|}$ be the matrix with entries δ_{ij} defined as

$$\delta_{ij} = \begin{cases} y_{(\omega_j)} - y_{(\omega_i)} & \text{if } \Pr(\omega_i \to \omega_j) > 0 \ \forall \omega_i, \omega_j \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

where $y_{(\omega_j)}$ is the height of configuration ω_i with respect to the reference line. It is easily verifiable that the diagonal elements of the matrix resulting from the product $\mathbf{P} \cdot \mathbf{Y}^{\mathrm{T}}$ are $h_1(\omega_0), h_1(\omega_1), \ldots, h_1(\omega_{|\Omega|})$ respectively. Defining the vector of length $|\Omega|$ from these diagonals as $\mathbf{h}_1 = \mathsf{diag}(\mathbf{P} \cdot \mathbf{Y}^{\mathrm{T}})$ and by comparing with equation (3.2), we arrive at the following recurrence

$$\mathbf{h}_2 = \mathbf{P} \cdot \mathbf{h}_1 + \mathbf{h}_1 = (\mathbf{P} + \mathbf{I}) \cdot \mathbf{h}_1$$
$$\mathbf{h}_3 = \mathbf{P} \cdot \mathbf{h}_2 + \mathbf{h}_1 = (\mathbf{P}^2 + \mathbf{P} + \mathbf{I}) \cdot \mathbf{h}_1$$

and more generally,

$$\mathbf{h}_t = \mathbf{P}\cdot\mathbf{h}_{t-1} + \mathbf{h}_1 = (\mathbf{P}^{t-1} + \ldots + \mathbf{P} + \mathbf{I})\cdot\mathbf{h}_1 = \sum_{k=0}^{t-1}\mathbf{P}^k\cdot\mathbf{h}_1 \tag{4.1}$$

Finding the expected change in height in $t \geq 1$ steps for a general system of particles starting in any configuration is then a simple matter of iteratively calculating matrix-vector products. As we commented in Section 3.2, for the two- and three-particle case, having that for some $N \in \mathbb{N}$ the vector $\mathbf{h}_N$ contains all positive values – a relation that we denote by $\mathbf{h}_N \succ 0$ – suffices. This is because $\mathbf{P}$ and its positive integer powers are row-stochastic matrices so that once $\mathbf{h}_N = \mathbf{P}\cdot\mathbf{h}_{N-1} + \mathbf{h}_1 \succ 0$, $\mathbf{h}_m \succ 0$ for all $m \geq N$.

However, and as we previously noted, in the general case, merely having that some vector $\mathbf{h}_N$ contains all positive values does not guarantee that the same will hold for $\mathbf{h}_m$ for all $m > N$. Rather, we need to verify that $h_t(\sigma_0)$ goes to infinity in the limit $t \to \infty$ for all starting configurations $\sigma_0 \in \Omega$. This brings us to our main lemma.

Lemma 4.1.1 *Let π be stationary distribution of the Markov chain corresponding to Algorithm 3. There exists an $N \in \mathbb{N}$ such that $\mathbf{h}_{m+1} - \mathbf{h}_m \succ 0$ for all integers $m \geq N$ if and only if $\pi\cdot\mathbf{h}_1 \succ 0$.*

Proof.

To prove the forward direction assume that such an N exists. Expanding the given inequality according to equation (4.1),

$$\mathbf{h}_{m+1} - \mathbf{h}_m = \left(\mathbf{P}^m + \mathbf{P}^{m-1} + \ldots + \mathbf{I}\right)\cdot\mathbf{h}_1 - \left(\mathbf{P}^{m-1} + \mathbf{P}^{m-2} + \ldots + \mathbf{I}\right)\cdot\mathbf{h}_1 = \mathbf{P}^m\cdot\mathbf{h}_1 \succ 0$$

Now recall that $\mathbf{P}$ is a (positive) row-stochastic matrix. So, left multiplying the above equation with powers of $\mathbf{P}$, we must also have $\mathbf{P}^{m+1}\cdot\mathbf{h}_1 \succ 0$, $\mathbf{P}^{m+2}\cdot\mathbf{h}_1 \succ 0$, … and so in the limit $m \to \infty$ we get $\lim_{m\to\infty}\mathbf{P}^m\cdot\mathbf{h}_1 = \pi\cdot\mathbf{h}_1 \succ 0$.

For the other direction, suppose that $\pi\cdot\mathbf{h}_1 \succ 0$. Then, using equation (4.1) and the definition of the stationary distribution we have

$$\lim_{m\to\infty}(\mathbf{h}_{m+1}-\mathbf{h}_m)=\pi\cdot\mathbf{h}_1\succ 0$$

So there exists $N_i\in\mathbb{N}$ such that for all $m\geq N_i$, for all configurations $\omega_i\in\Omega$, ($i=1,2,\ldots,|\Omega|$) and for every $\epsilon>0$, the following is true

$$\left|h_{m+1}(\omega_i)-h_m(\omega_i)-\pi_{\omega_i}h_1(\omega_i)\right|<\epsilon$$

or equivalently,

$$\pi_{\omega_i}h_1(\omega_i)-\epsilon<h_{m+1}(\omega_i)-h_m(\omega_i)<\pi_{\omega_i}h_1(\omega_i)+\epsilon$$

Since by assumption, $\pi_{\omega_i}h_1(\omega_i)>0$, we choose $\epsilon=\frac{\pi_{\omega_i}h_1(\omega_i)}{2}>0$. By going further enough out in the sequence, we must still have a natural number N_i' with $m\geq N_i'$ satisfying,

$$h_{m+1}(\omega_i)-h_m(\omega_i)>\pi_{\omega_i}h_1(\omega_i)-\epsilon=\pi_{\omega_i}h_1(\omega_i)-\frac{\pi_{\omega_i}h_1(\omega_i)}{2}>0$$

Re-writing in vector form, for all $m\geq N$ we have $\mathbf{h}_{m+1}-\mathbf{h}_m\succ 0$ for $N=\max_i N_i'$.

□

Given the conditions of Lemma 4.1.1, we obtain the following corollary

Corollary 4.1.1 *If $\pi\cdot\mathbf{h_1}\succ 0$, then starting in any configurations $\sigma_0\in\Omega$, $\lim_{t\to\infty}h_t(\sigma_0)$.*

Proof.

From equation (4.1), $\mathbf{h}_t=\sum_{k=0}^{t-1}\mathbf{P}^k\cdot\mathbf{h}_1$. So, in the limit $t\to\infty$,

$$\lim_{t\to\infty}\mathbf{h}_t=\sum_{k=0}^{\infty}\mathbf{P}^k\cdot\mathbf{h}_1$$

Now suppose that stationarity is reached at some step, $T<\infty$

$$\lim_{t\to\infty} \mathbf{h}_t = \sum_{k=T}^{\infty} \pi \cdot \mathbf{h}_1 + \sum_{k=0}^{T-1} \mathbf{P}^k \cdot \mathbf{h}_1 \tag{4.2}$$

According to Lemma 4.1.1, $\pi \cdot \mathbf{h}_1 \succ 0$. So the first sum in equation (4.2) goes to (a vector with all values) infinity. Therefore, $\lim_{t\to\infty} h_t(\sigma_0) = \infty$ for all $\sigma_0 \in \Omega$.

□

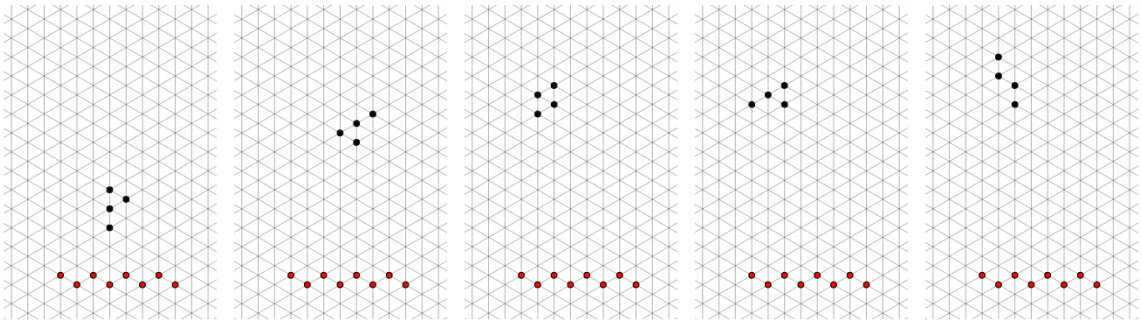

Figure 4.1: Phototaxing simulation in a system of four amoebots.

This leads us to a simple algorithm for verifying phototaxing in arbitrarily large systems. Using Algorithm 4, obtain an estimate $\tilde{\mathbf{P}}$ for the staionary distribution π and verify whether $\tilde{\mathbf{P}} \cdot \mathbf{h}_1 = \tilde{\mathbf{P}} \cdot \mathsf{diag}(\mathbf{P} \cdot \mathbf{Y}^{\mathrm{T}}) \succ 0$. If the answer is yes, then from Corollary 4.1.1 we know that phototaxing occurs in expectation; if the answer is no, then we would have disproved the conjecture of [29] regarding expected phototaxing in the general case.

Algorithm 4 Approximating the stationary distribution π

Given: $\mathbf{P}$ as defined in Section 4.1

$\mathbf{P}' \leftarrow \mathbf{P}$

$\tilde{\mathbf{P}} \leftarrow \mathbf{P}$

$\epsilon \leftarrow 10^{-}4$ ▷ error tolerance

while $\left\{|\tilde{p}_{ij} - p'_{ij}| \geq \epsilon : \tilde{p}_{ij} \in \tilde{\mathbf{P}}, p'_{ij} \in \mathbf{P}'\right\}$ **do**

    $\mathbf{P}' \leftarrow \tilde{\mathbf{P}}$

    $\tilde{\mathbf{P}} \leftarrow \tilde{\mathbf{P}} \cdot \mathbf{P}$

return $\tilde{\mathbf{P}}$

4.2 Expected phototaxing for the four particle system

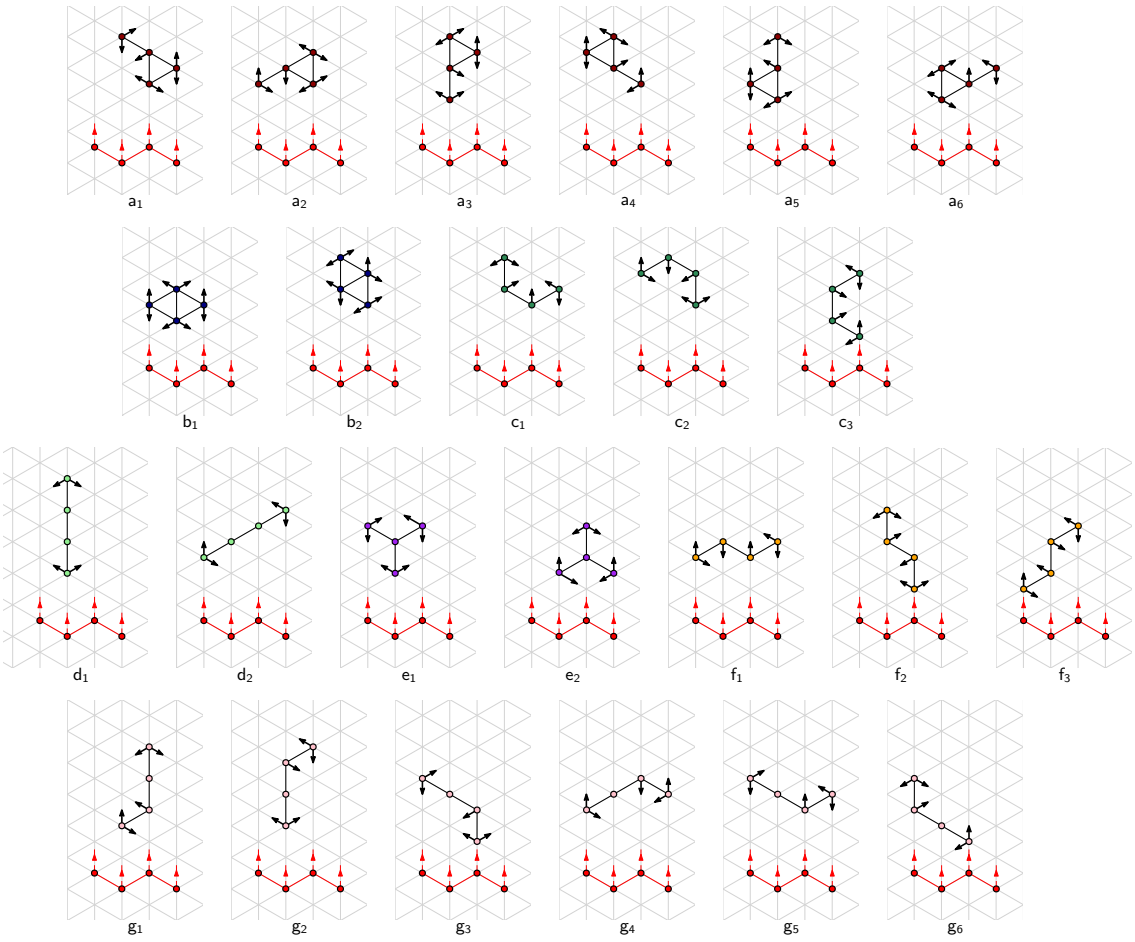

Figure 4.2: All unique states in a four-particle system and the valid moves for each configuration. Colour of the particles indicate the class to which the configuration belongs.

Following the steps described in the previous section, we are able to verify expected phototaxing for two- and three-particle systems. Figure 4.2 illustrates all the unique states for the four-particle system. We divide the configurations into seven equivalence classes that are indicated by the colour of the particles in the figure and also by the alphabet of their labels. All configurations in the same equivalence class are obtained by rotation with respect to G_Δ. It should be noted that these configurations are unique upto rotation and translation, and in fact, any two configurations in the same equivalence class and with the same probabilities of transition (with respect to the direction of external signal) are mutually indistinguishable. In Figure 4.3, we see two configurations from class *c* that are

indistinguishable with respect to the external signal.

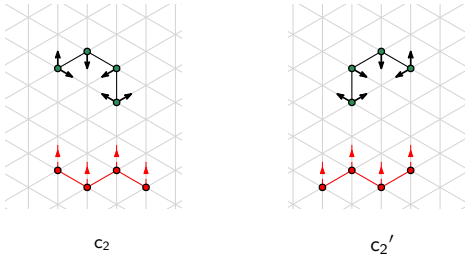

Figure 4.3: c_2 and c_2' are indistinguishable with respect to the signal.

As we demonstrate in Figure 4.4, the height of the system becomes positive in a finite number of transitions for $\lambda = 4, 5, 6$ in two-, three- and four-particle systems. Phototaxing is also experimentally observed (see figure 4.1) for the four-particle case. As a result, we have the following theorem:

Theorem 4.2.1 *For systems up to size four, phototaxing occurs in expectation.*

Further, Figure 4.6 gives all the possible transitions from fixed states, along with corresponding transition probabilities and $h_1(\cdot)$ values. The computation of these values can be done manually by referring to the figure; an example is also provided in Subsection 4.2.1.

The proof follows from correctness of our verification algorithm that allows us to check that $\tilde{\mathbf{P}} \cdot \mathbf{h}_1 > 0$ for the four-particle case for all $\lambda \geq 4$. We note that the phototaxing verification algorithm requires pre-calculating the transition matrix $\mathbf{P}$ and the change in height $\mathbf{Y}$. Given that the number of configurations grows exponentially with the size of the system [31], this turns out to be a non-trivial requirement and prevents the algorithm from scaling. Any formal proof of phototaxing in the general case is therefore unlikely to be obtained directly from this technique. Instead, it offers a concise definition of phototaxing in terms of the stationary distribution of its corresponding Markov chain.

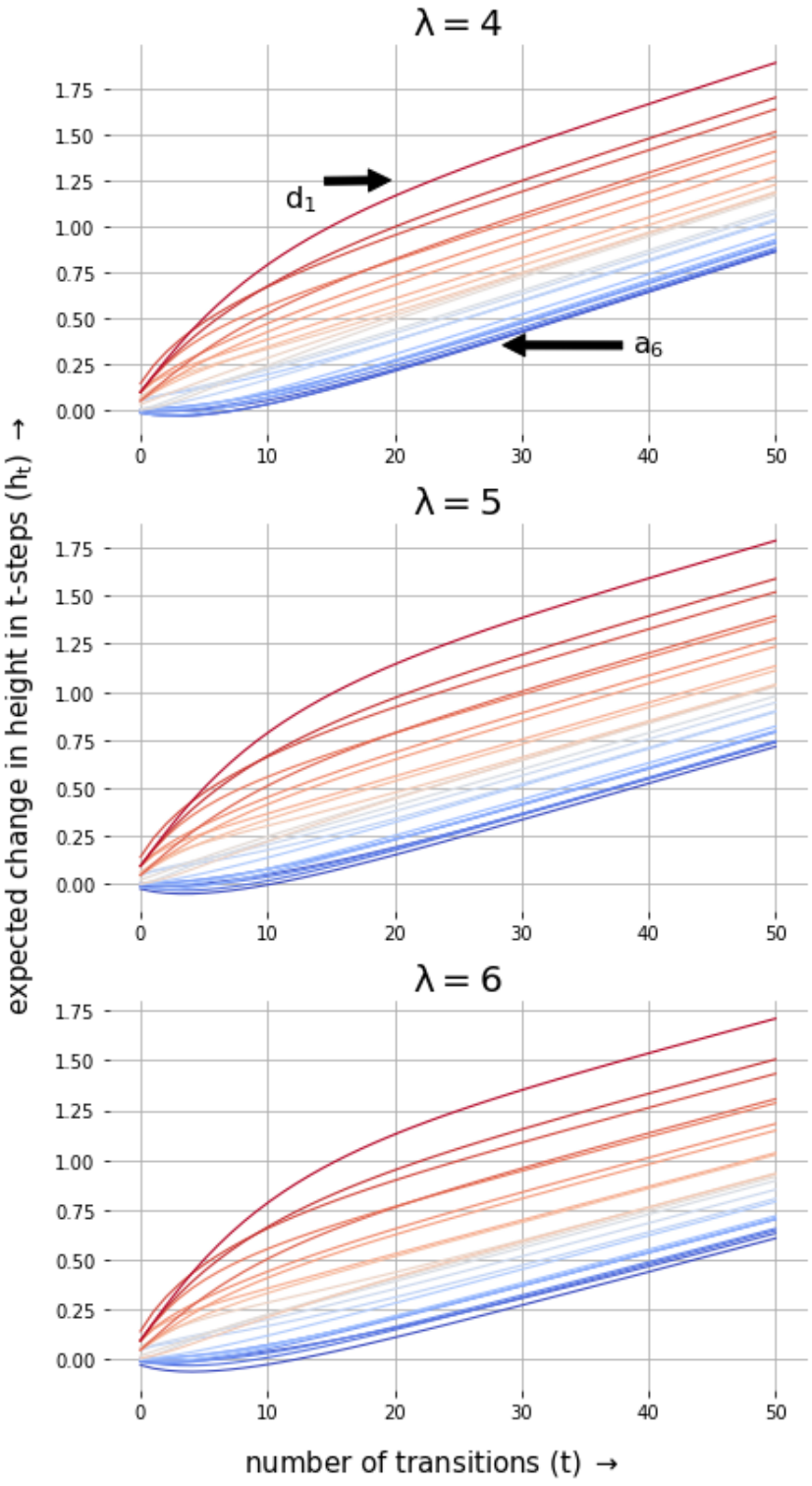

Figure 4.4: The t-step expected change in height values plotted for all possible configurations in the four-particle case. Configurations corresponding to the highest (d_1) and lowest (a_6) overall values are also shown.

4.2.1 Calculating the function h_1

The function $h_1 : \Omega \to \mathbb{Q}$ gives the one-step change in height for a system configuration of given size. We briefly explain how to calculate this value for a general system using equation (3.1) and Figure 4.6. For any given star graph, multiply the probability corresponding to the colour on an edge with the label on the edge representing the change in height for that edge/transition. Repeat this for every edge, including double edges. The resulting sum is the value $\sum_{\sigma_1 \in \Omega}(y_1 - y_0)\Pr(\sigma_0 \to \sigma_1) = \mathbb{E}[\Delta_1 y \mid \sigma_0]$. An example is given for configuration b_2 in Figure 4.5.

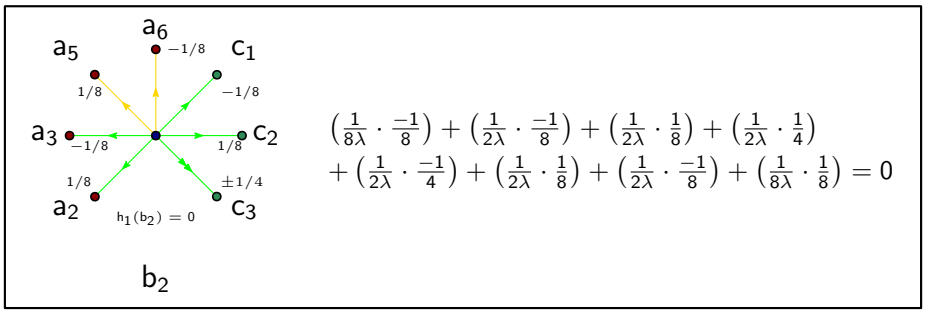

Figure 4.5: Calculation of $h_1(b_2)$ starting from the top-most edge and proceeding clockwise.

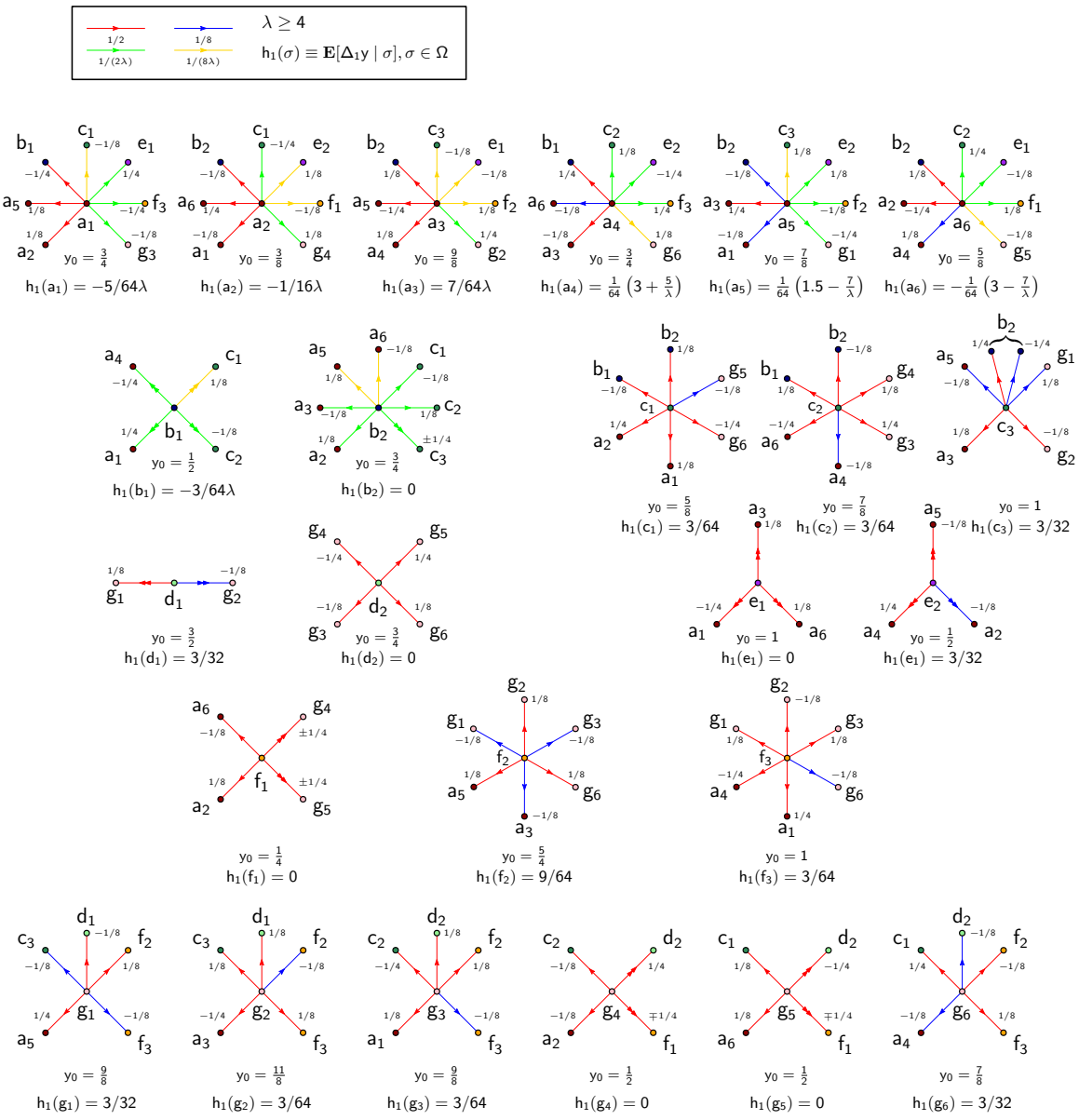

Figure 4.6: All possible state transitions, in the direction of the arrow, with respect to the configurations shown in Figure 4.2. colours indicate the probability of transition (see legend) and labels give the resulting change in height due to that transition. Double arrows to a configuration indicate that there are two ways to reach it. y_0 gives the center-of-height of the system and $h_1(\cdot)$ its expected change in height in one step.

Chapter 5

Stochastic Maze Solving

The nearly-stateless nature of stochastic algorithms for the amoebot model makes it a highly desirable approach for solving a variety of problems of movement and configuration within the model. This is a useful property to have for maze-solving where deterministic algorithms might require careful definitions for system behaviour based on the obstacles encountered by the particles. In this chapter we describe the problem of maze-solving and propose a Markov chain based algorithm for maze-solving under the geometric amoebot model. Our algorithm exploits the results from phototaxing concerning asymmetric activation of particles and introduces adaptive variation in the compression parameter to control the system's behaviour as it drifts through the maze.

Maze solving. In automated maze-solving, we are given a maze or a network of paths, with one or more paths leading to a (typically) unique exit. While there are multiple ways of stating the problem, we say that a maze is solved when an agent, starting at any location in the maze, is able to find a path to the exit. Within the context of the amoebot model, we consider mazes to be made up of a collection of vertices forming an open-polygon on the triangular lattice, with the opening forming the exit. We call such vertices, *walls* of the maze. It must also be ensured that a particle does not expand onto a wall.

5.1 An Algorithm for Maze Solving

Our first step will be to extend the ability of individual particles to send a ray (or signal) up to a fixed distance, d along the lattice lines. Rays from a particle are reflected back by the walls, allowing the particle to calculate its distance from that wall. Since the particles

themselves do not reflect the rays but only generate them, they are virtually indistinguishable and the anonymity property is not violated. Further, the restriction on the distance d allows us to bound the memory required by individual particles – any signal that is not reflected within distance d simply vanishes – even though the triangular lattice may theoretically be unbounded.

At a high level, the algorithm behaves in the following manner. Each particle carries an internal temperature τ. Rays reflected by walls at a (edge) distance $k \leq d$ from the emitting particle, upon returning, transfer an additional $\tau_0 \gamma^{-k}$ amount of heat to the particle for parameters τ_0 (unbounded) and $\gamma \in (0, 1]$. On activation, each particle in the configuration, σ, will choose a direction, uniformly at random, to move. The particle will then send a ray up to at most a distance d in the chosen direction and adjust its probability of movement in proportion to the weight function,

$$w(\sigma, k) = \left(e^{1/T(k)}\right)^{\varepsilon(\sigma)}$$

where $T(k) = \tau + \tau_0 \gamma^{-k}$ is the *temperature function*, and $\varepsilon(\sigma)$ gives the number of edges in the current configuration σ. We set the parameters such that when an activating particle is closer to the walls, lower values for the temperature function are achieved and the system performs net compression, and conversely when it is farther away, higher temperature results in net expansion of the system. In this manner we replace the compression parameter λ with the exponential function $e^{1/T(k)}$ that depends on the particle's distance from the wall. Figure 5.1 plots the evolution of the exponential function for different values of k.

Algorithm 5 Maze-solving subroutine of each particle $\mathcal{P}$.

Given: d, τ_0, τ **and** γ

1: $\mathcal{P}$ picks a direction uniformly at random.
2: It then sends a ray up to a maximum distance d in the chosen direction.
3: **if** ray returns to $\mathcal{P}$ after travelling a total distance of $2k$ **then**
4: $\quad$ $\mathcal{P}$ sets $\lambda \leftarrow \exp\left(\frac{1}{\tau + \tau_0 \gamma^{-k}}\right)$.
5: **else** ▷ ray does not return to $\mathcal{P}$
6: $\quad$ $\mathcal{P}$ sets $\lambda \leftarrow 1$.
7: It executes steps 2–8 of Algorithm $\mathcal{MC}_{\text{compress}}$ with parameter value λ.

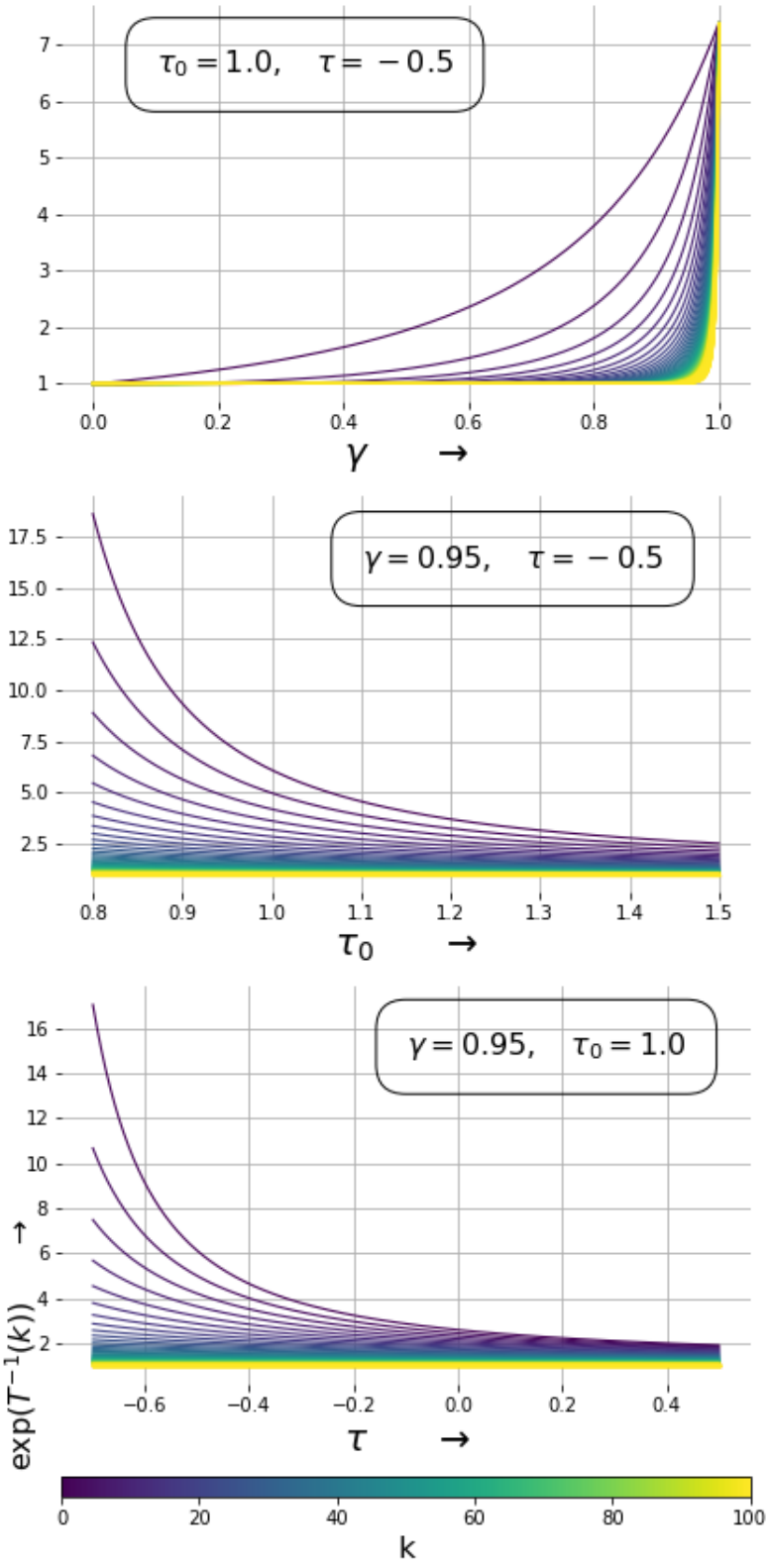

Figure 5.1: The exponential function $e^{1/T(k)}$ plotted for a range of values of γ, τ_0 and τ while keeping the other two values constant.

As in the phototaxing algorithm [29], if we were to set the particle activation rates (which are currently uniform) with respect to the distance from the walls instead of the weights, maze-solving algorithm for each particle would simply be Algorithm $\mathcal{MC}_{\text{compress}}$ with the new activation rates.

5.2 Pymoebots: demonstrable compression and phototaxing

To experimentally test our algorithm, we developed a virtual simulation environment. The *Pymoebots* simulator exposes a JavaScript based graphical-interface (figure 5.2) hosted on a local server running a Python back-end. The minimalistic interface provides some basic controls for setting up custom experiments by placing model objects (walls and amoebots) on the triangular lattice, loading and saving experiments and controlling playback of loaded experiments.

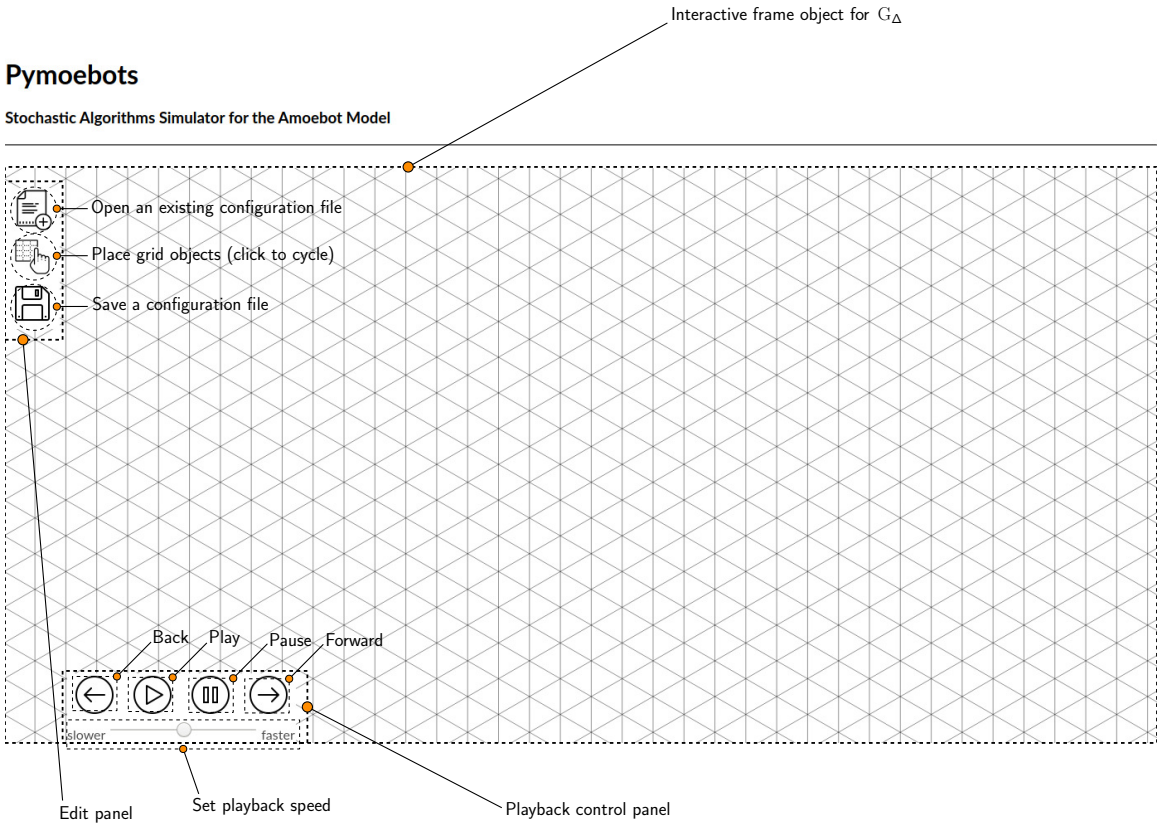

Figure 5.2: The Pymoebots simulator's graphical user interface.

While the software project continues to be under development, the Pymoebots simulator is currently capable of running, in addition to our maze-solving algorithm, the algorithms for compression [5] and phototaxing [29] as we have shown in figures 3.1 and 4.1. Although, the software-package is intended to provide functional modules for stochastic algorithms under the geometric amoebot model, it is written with a more general SOPS framework in mind. With an eye towards creating a self-contained software-package for SOPS algorithms, we present a preliminary version of our project: `pymoebots-base`.

As an alternative, Daymude et al. [7] have open-sourced a deterministic algorithms simulator with the ability to add custom experimental modules including those for stochastic algorithms. While Pymoebots was developed independently of there simulator, we cite them here for completeness.

5.3 Experiments

We now provide evidence through simulations that systems of amoebots demonstrate maze-solving in three classes of mazes and certain combinations thereof. We also show specific instances where Algorithm 5 does not lead to a path to the exit. Analysing these instances helps us understand the shortcomings of the proposed algorithm and suggests an alternative direction to consider for the problem of maze-solving within the amoebot model.

We look at three basic classes of mazes that are illustrated in Figure 5.3. Namely, these are *forks*, *bends* and *cavities*. The *bottleneck width*, or simply width, of a maze is the width of the narrowest part of the maze. The length of the shortest path over G_Δ starting from the particle system's center-of-height to the exit gives the system's *exit distance*. The size of a system, the bottleneck width of a maze and the exit distance are parameters that characterise each experiment. Having a uniform set of experimental parameters allows us to talk about different types mazes in a consistent way.

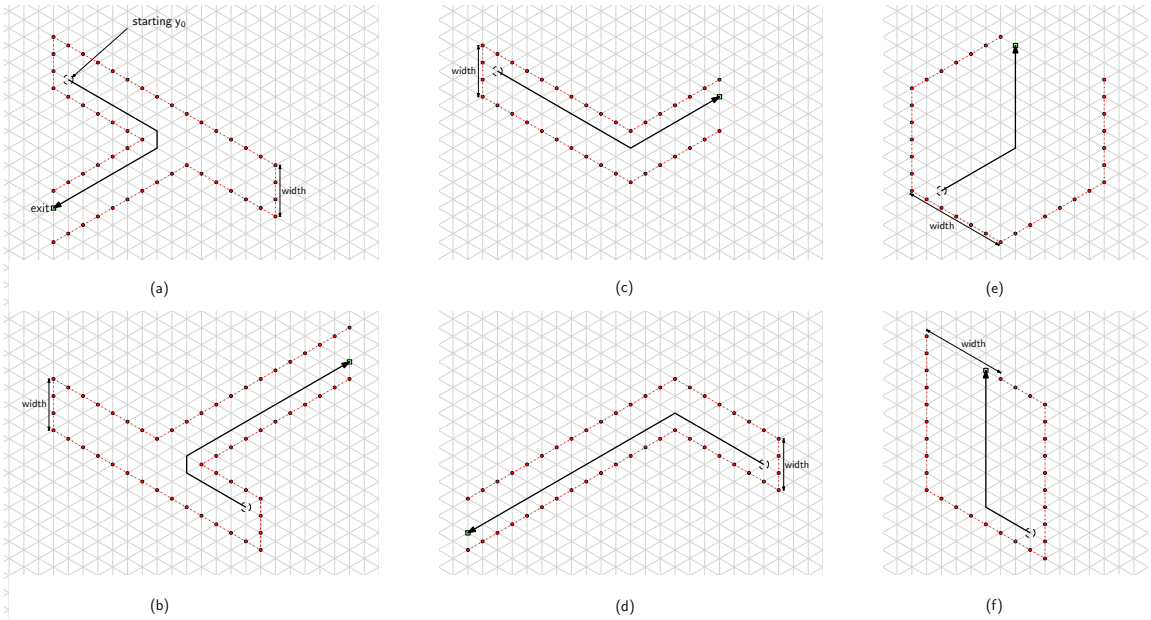

Figure 5.3: Examples of the three classes of mazes: (a and b) forks; (c and d) bends; and (d and e) cavities. The bold black lines indicate the shortest path to the exit for each maze.

5.3.1 Results

We observe in Figure 5.4 that each of the three types of mazes is solved by a system of amoebots. The top-most simulation comprises of a system of 15 particles in a fork-shaped maze of width 7, exit distance 50. The simulation in the middle is performed in a bend-shaped maze of width 6, exit distance 25 by a system of 7 particles, and the bottom-most simulation has the same parameters for a cavity.

We also performed experiments on various combinations of the three classes of mazes. As seen in Figure 5.5, the system eventually solves the two combinations of mazes as well.

As we previously stated, Algorithm 5 does not always find a path to the exit. Now consider the two setups in Figure 5.6. In both cases the system remains trapped in the circled regions even after running each simulation for 3 million iterations. Compare the simulations in figures 5.4 (b) and 5.6 (b): while the former was successfully solved, a system of 15 particles could not solve the latter of the two bend-shaped mazes of width 7, exit distance 50 over repeated simulations. The flaw in our maze-solving algorithm is that it relies on the maze to guide the system to the exit. We speculate that, because a system in the circled region is at the same distance from the opposite walls of the maze,

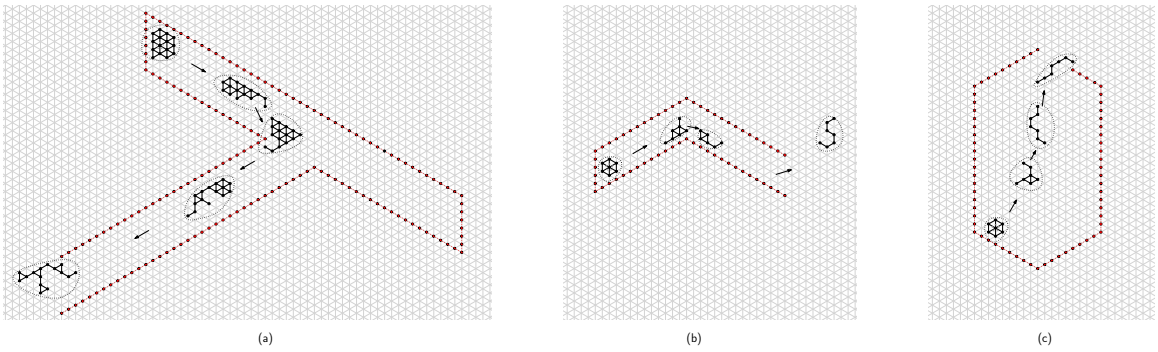

Figure 5.4: Each of the three types of mazes solved by a particle system. (a) A fork solved by a system of 15 amoebots in approximately 100,000 iterations.(b) a bend solved by a system of 7 amoebots in approximately 30,000 iterations; and (c) a cavity solved by a system of 7 amoebots in 60,000 iterations.

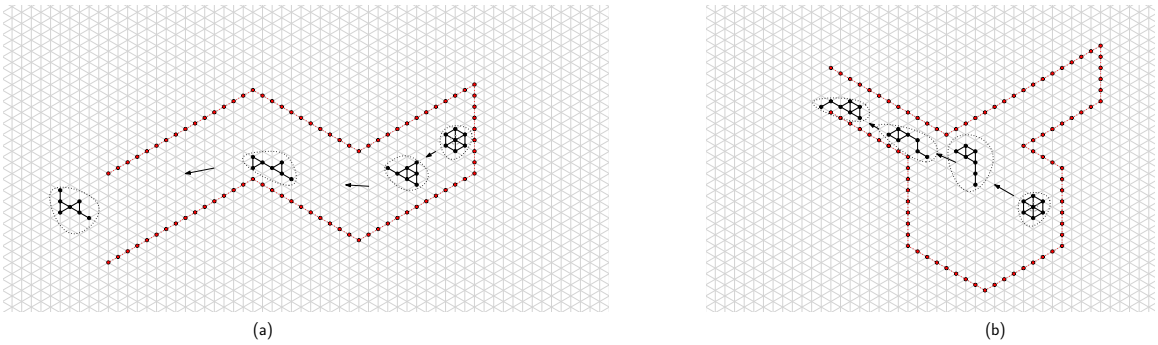

Figure 5.5: Examples of combinations of mazes solved by particle systems. We see (a) a combination of a bend and cavity solved by a system of 7 amoebots in approximately 20,000 iterations; (b) a combination of bends solved by a system of 7 amoebots in approximately 45,000 iterations.

the random-walk is effectively trapped and the effect gets more pronounced as the exit distance is increased.

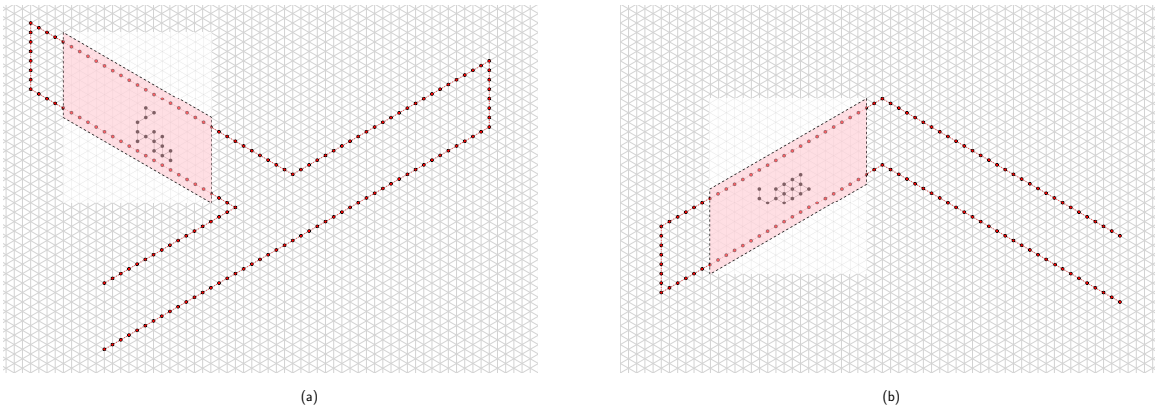

Figure 5.6: Instances where the maze-solving algorithm fails.

Recall the example of navigation by ants from Section 1.2 – to navigate unfamiliar territories, ants create a network of chemical trails that lead to the destination through gradual reinforcement of the optimal path [30]. In contrast, our algorithm for maze-solving is memory-less in that a system has no way of knowing if it has been at a certain location before. This motivates us to consider a more general, ant-inspired algorithm for maze-solving wherein the particles are able to leave trails on the triangular lattice to help guide other particles to a path out of the maze. We continue to develop an optimal approach to stochastic maze-solving in the geometric amoebot model and list this as a possible direction in our future work.

Chapter 6

Conclusion

In this work we gave an algorithm for verifying that phototaxing occurs in expectation. We then used this algorithm to extend the previous result of expected phototaxing from particle systems of size three to those of size four. The algorithm for phototaxing verification gives us a way to think about the phenomenon of phototaxing in terms of the stationary distribution of the associated Markov chain. It would be interesting to see if this intuition can be utilised to arrive at a general proof or a counter-example for expected phototaxing in systems of all sizes.

We also defined the problem of maze-solving under the geometric amoebot model and proposed an algorithm that solves certain types of mazes. Our algorithm for maze-solving works by varying the compression parameter based on the distance to the walls, as a result the individual particles oscillate between compression and expansion to navigate the maze via a random-walk. Using a simulator that we developed, we then experimentally demonstrated maze-solving ability of amoebot systems for various special types of mazes and combinations thereof. Through experimentation, we also identified instances of mazes where the amoebot system does not solve the maze, often getting trapped the same region. Noting the shortcomings of our maze-solving algorithm, an ant-inspired algorithm seems to be a plausible approach to collective maze-solving in the amoebot model. Unlike amoebots in the compression algorithm (and its variants), ants are not tethered to each other. The recent work of Li et al. [23] presents a new algorithm for compression that does not require the connectivity constraint. This opens up several exciting possibilities for ant-inspired algorithms in the amoebot model.

Most new work in stochastic algorithms for the amoebot model has evolved from the

algorithms for compression and expansion. It would therefore be in our interest to identify a more principled approach to adjust the Markov chain parameters, specifically the objective function, such that the system oscillates between compression and expansion, allowing for phototaxing and maze-solving to be viewed as emergent behaviour.

The geometric amoebot model provides a rigorous theoretical framework for the study of self-organizing particle systems. Considering that this is a relatively new model, there are several unexplored problems to be solved in both theoretical as well as applied domains.

References

[1] Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(11):1021–1024, November 1994.

[2] Marta Andrés Arroyo, Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A stochastic approach to shortcut bridging in programmable matter. *Natural Computing*, 17(4):723–741, Dec 2018.

[3] Dan Boneh, Christopher Dunworth, Richard J. Lipton, and JirÌĘÃŋ Sgall. On the computational power of dna. *Discrete Applied Mathematics*, 71(1):79–94, 1996.

[4] Sarah Cannon, Joshua J. Daymude, Cem Gökmen, Dana Randall, and Andréa W. Richa. A local stochastic algorithm for separation in heterogeneous self-organizing particle systems. In Dimitris Achlioptas and Laszlo A. Vegh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019*, Leibniz International Proceedings in Informatics, LIPIcs. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, September 2019. 22nd International Conference on Approximation Algorithms for Combinatorial Optimization Problems and 23rd International Conference on Randomization and Computation, APPROX/RANDOM 2019 ; Conference date: 20-09-2019 Through 22-09-2019.

[5] Sarah Cannon, Joshua J. Daymude, Dana Randall, and Andréa W. Richa. A markov chain algorithm for compression in self-organizing particle systems. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 279–288, New York, NY, USA, 2016. Association for Computing Machinery. A significantly detailed journal version is available on arXiv:1603.07991.

[6] Joshua J. Daymude, Zahra Derakhshandeh, Robert Gmyr, Alexandra Porter, Andréa Richa, Christian Scheideler, and Thim Strothmann. On the runtime of universal coating for programmable matter. *Natural Computing*, 17(1):81–96, March 2018. Funding Information: Joshua J. Daymude, Zahra Derakhshandeh, Alexandra Porter, Andréa W. Richa: Supported in part by NSF Grants CCF-1353089, CCF-1422603, and REU–026935. Robert Gmyr, Christian Scheideler, Thim Strothmann: Supported in part by DFG Grant SCHE 1592/3-1.

[7] Joshua J. Daymude, Robert Gmyr, and Kristian Hinnenthal. AmoebotSim: A Visual Simulator for the Amoebot Model of Programmable Matter. Available online at https://github.com/SOPSLab/AmoebotSim, 2021.

[8] Joshua J. Daymude, Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Christian Scheideler, and Andréa W. Richa. Convex hull formation for programmable matter. In *Proceedings of the 21st International Conference on Distributed Computing and Networking*, ICDCN 2020, pages 2:1–2:10, 2020.

[9] Joshua J. Daymude, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Improved leader election for self-organizing programmable matter. In Antonio Fernández Anta, Tomasz Jurdzinski, Miguel A. Mosteiro, and Yanyong Zhang, editors, *Algorithms for Sensor Systems*, pages 127–140, Cham, 2017. Springer International Publishing.

[10] Joshua J. Daymude, Kristian Hinnenthal, Andréa W. Richa, and Christian Scheideler. *Computing by Programmable Particles*, pages 615–681. Springer International Publishing, Cham, 2019.

[11] Joshua J. Daymude, Andréa W. Richa, and Jamison W. Weber. Bio-inspired energy distribution for programmable matter. In *International Conference on Distributed Computing and Networking 2021*, ICDCN '21, pages 86–95, New York, NY, USA, 2021. Association for Computing Machinery.

[12] Zahra Derakhshandeh, Robert Gmyr, Andréa Richa, Christian Scheideler, and Thim Strothmann. Universal coating for programmable matter. *Theoretical Computer Science*, 671:56–68, April 2017. Publisher Copyright: © 2016 Elsevier B.V. Copyright: Copyright 2017 Elsevier B.V., All rights reserved.

[13] Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. An algorithmic framework for shape formation problems in self-organizing particle systems. In *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication*, NANOCOM' 15, New York, NY, USA, 2015. Association for Computing Machinery.

[14] Zahra Derakhshandeh, Robert Gmyr, Andrea W. Richa, Christian Scheideler, and Thim Strothmann. Universal shape formation for programmable matter. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '16, pages 289–299, New York, NY, USA, 2016. Association for Computing Machinery.

[15] Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida Bazzi, Andréa W. Richa, and Christian Scheideler. Leader election and shape formation with self-organizing programmable matter. In Andrew Phillips and Peng Yin, editors, *DNA Computing and Molecular Programming*, pages 117–132, Cham, 2015. Springer International Publishing.

[16] Zahra Derakhshandeh, Andréa Richa, Shlomi Dolev, Christian Scheideler, Robert Gmyr, and Thim Strothmann. Brief announcement: Amoebot-a new model for programmable matter. In *SPAA 2014 - Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, Annual ACM Symposium on Parallelism in Algorithms and Architectures, pages 220–222. Association for Computing Machinery, January 2014. 26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2014 ; Conference date: 23-06-2014 Through 25-06-2014.

[17] Shlomi Dolev, Robert Gmyr, Andréa W. Richa, and Christian Scheideler. Amebot-inspired self-organizing particle systems. *CoRR*, abs/1307.4259, 2013.

[18] David Doty. Theory of algorithmic self-assembly. *Commun. ACM*, 55(12):78–88, December 2012.

[19] R. Gmyr. Distributed algorithms for overlay networks and programmable matter. 2018. Ph.D. thesis, Paderborn University.

[20] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[21] Ernst Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, Feb 1925.

[22] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society Providence, RI, USA., 2009.

[23] Shengkai Li, Bahnisikha Dutta, Sarah Cannon, Joshua J. Daymude, Ram Avinery, Enes Aydin, Andréa W. Richa, Daniel I. Goldman, and Dana Randall. Programming active cohesive granular matter with mechanically induced phase changes, 2020.

[24] Fabio Martinelli and Fabio Lucio Toninelli. On the mixing time of the 2d stochastic ising model with "plus" boundary conditions at low temperature. *Communications in Mathematical Physics*, 296(1):175–213, May 2010.

[25] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[26] Toshiyuki Nakagaki, Hiroyasu Yamada, and Ágota Tóth. Maze-solving by an amoeboid organism. *Nature*, 407(6803):470–470, Sep 2000.

[27] Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13:195–224, 2013.

[28] Chris R. Reid and Tanya Latty. Collective behaviour and swarm intelligence in slime moulds. *FEMS Microbiology Reviews*, 40(6):798–806, 08 2016.

[29] William Savoie, Sarah Cannon, Joshua J. Daymude, Ross Warkentin, Shengkai Li, Andréa W. Richa, Dana Randall, and Daniel I. Goldman. Phototactic supersmarticles. *Artif. Life Robot.*, 23(4):459–468, December 2018.

[30] D.J.T Sumpter. The principles of collective animal behaviour. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 361(1465):5–22, Jan 2006.

[31] Markus Vöge and Anthony J. Guttmann. On the number of hexagonal polyominoes. *Theoretical Computer Science*, 307(2):433–453, 2003. Random Generation of Combinatorial Objects and Bijective Combinatorics.

[32] Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 353–354, New York, NY, USA, 2013. Association for Computing Machinery.