A Symbiotic Perspective on Distributed Algorithms and Social Insects

by

Tsvetomira Radeva

B.S., State University of New York, College at Brockport (2010)
S.M., Massachusetts Institute of Technology (2013)

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

© Massachusetts Institute of Technology 2017. All rights reserved.

Author .....
Department of Electrical Engineering
and Computer Science
March 17, 2017

Certified by .....
Nancy Lynch
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chair of the Committee on Graduate Students

A Symbiotic Perspective on Distributed Algorithms and Social Insects

by

Tsvetomira Radeva

Submitted to the Department of Electrical Engineering
and Computer Science
on March 17, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Biological distributed algorithms are decentralized computer algorithms that solve problems related to real biological systems and provide insight into the behavior of actual biological species. The biological systems we consider are social insect colonies, and the problems we study include foraging for food (exploring the colony's surroundings), house hunting (reaching consensus on a new home for the colony), and task allocation (allocating workers to tasks in the colony). The goal is to combine the approaches used in understanding complex distributed and biological systems in order to develop (1) more formal and mathematical insights about the behavior of social insect colonies, and (2) new techniques to design simpler and more robust distributed algorithms. Our results introduce theoretical computer scientists to new metrics, new ways to think about models and lower bounds, and new types of robustness properties of algorithms. Moreover, we provide biologists with new tools and techniques to gain insight and generate hypotheses about real ant colony behavior.

Thesis Supervisor: Nancy Lynch
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I would like to thank my advisor, Nancy Lynch, for introducing me to biological distributing algorithms and showing me what it takes to make progress in a new area. From Nancy, I learned how to distill the clearest arguments and spot the key intricacies in proofs. I would also like to thank Radhika Nagpal and Nir Shavit for being on my thesis committee and helping me understand the wider implications of the work presented in this thesis.

Many of the results in this thesis would not have been possible without the help of my collaborators: Cameron Musco, Christoph Lenzen, Hsin-Hao Su, Anna Dornhaus, Radhika Nagpal, Mohsen Ghaffari, and Calvin Newport. They have been my best teachers in the past few years, always willing to work on new problems, and always ready to read more drafts.

Last but not least, my family and friends have been extremely supportive and understanding, despite the friendly reminders that I have to graduate eventually. Special mentions go out to Srikanth, who can no longer win arguments by simply saying "Trust me, I'm a doctor", and to baby Arlen, for his cooperation in writing this thesis.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Biological Distributed Algorithms

Recent works in the distributed computing and biology communities have established parallels between the problems studied in theoretical computer science and behavioral ecology. This gave rise to the area of *biological distributed algorithms*: decentralized computer models and algorithms that solve problems related to real biological systems and provide insight into the behavior of actual biological species. The biological systems range from simple cells [1], to slime mold [17], to social insect colonies [28, 50, 51, 76, 83], and the problems are usually well-known theoretical computer science problems like constructing a maximal independent set [1], finding shortest paths in a graph [17], task allocation [28], and graph exploration [50, 51, 76].

The key features of both the distributed and biological systems mentioned above are very similar and common to all self-organizing systems [19, 20, 64]: no central control that instructs the individuals, common global goal for all individuals to achieve, potentially limited communication and computation capabilities of the individuals, various degrees of noise and failures. A global goal and lack of centralized control are the defining features of most distributed systems. In social insect colonies, despite the presence of a queen, individuals are also rarely instructed what action to perform next. Moreover, there are numerous examples of common global goals that social insect colonies need to achieve in order to guarantee the survival of the colony: building

a nest, taking care of the brood items, or foraging for food to feed the entire colony.

Communication is beneficial and often necessary to solve many problems in distributed computing, and is also present in many biological systems. The two main types of communication employed by distributed algorithm designers are shared memory and message passing. In shared memory, individuals can read and write values at a common location (set of memory registers), while in message passing, individuals send messages to each other over a set of communication channels. In social insect colonies, similar, although constrained, modes of communication are also used. The shared-memory communication equivalent in social insect colonies is usually referred to as *stigmergy*: a mechanism to sense the environment and infer the need to perform a task. For example, ants and bees can sense the temperature in the nest, sense that larvae need to be fed, or follow a pheromone trail from the nest leading to a food source. The message-passing communication equivalent in social insect colonies are (random) ant-to-ant interactions through which ants share limited information by sensing each other's pheromones. Furthermore, such random interactions between agents is also very common in some distributed computing models like population protocols [6, 8] (discussed in more detail in Section 1.2.2).

Finally, various levels of noise and failures are present in both distributed systems and social insect colonies. In distributed systems, the standard assumptions involve upper bounds on the number of computing devices that may crash at any time, where crashes may be of several flavors depending on their severity and possibility of recovery. In social insect colonies, as in any biological system, failures are common and inevitable. Moreover, individuals are believed to act using noisy sensory information and potentially imprecise knowledge of the environment and colony parameters (such as colony size and amount for work needed).

Despite the numerous similarities between distributed and biological systems, they differ in many subtle but important aspects. Clearly, a key difference between these two types of systems is that the former is man-made and designed to achieve well-specified goals, while the latter is naturally-evolving with largely unknown specifications. For example, in distributed systems, there are usually assumptions on the total

number of processes, the maximum number of failures among them, the speed with which they take steps, and the ways in which they can interact with each other. The algorithms that run on these distributed systems are also specified exactly as either pseudocode or real executable code. Based on these assumptions and algorithms, it is usually possible to predict and analyze the behavior of the system in terms of the time it requires to achieve a certain state, or the correctness and precision with which the system reaches such a state. In biological systems, on the other hand, many of these assumptions and algorithms are not well-defined or completely unknown.

In biological systems, similarly to computer systems, there are numerous attempts to study the behavior of certain species analytically, both using centralized [15, 16, 79, 110] and distributed approaches [41, 60, 86, 95]. In contrast to distributed systems, the mathematical results that describe the behavior of biological systems are less rigorous and precise, often focusing on some tendency of the colony to converge to a specific state, but rarely analyzing specific metric functions of the key colony parameters. Also in contrast to distributed systems, biological systems and models are much more tolerant to noise and uncertainty. For example, many social insect colonies (and their corresponding biological models) are capable of surviving and even thriving under extreme conditions, while distributed systems (in theory or in real applications) often suffer fatal errors even in mildly unfavorable settings.

Many of the tools and techniques from distributed computing can be useful in understanding the behavior of social insect colonies more formally and quantitatively. Generally speaking, formal mathematical models can help abstract away from the complex biological world and yield a feasible computational analysis of the algorithms social insects use. In particular, models in distributed computing are abstract, discrete, probabilistic, and modular [10, 78, 89]; each individual is modeled independently from other individuals and from the environment. In these models, each individual is assumed to run an independent copy of a distributed algorithm. The resulting behavior of the individuals is analyzed using proof techniques from probability theory and algorithm complexity, to derive provable guarantees on the solvability and efficiency of the problems and algorithms, respectively.

Similarly, the biological approaches, both analytical and experimental, to understanding complex systems can be beneficial to designing and analyzing more robust distributed algorithms. When biologists study the behavior of social insect colonies, they focus on theoretical (mathematical) models [15, 41, 60, 79, 95] and practical experiments [21, 29, 91, 103, 104]. The goal of the experiments is usually to form and practically validate hypotheses about specific aspects of the insects' behavior. The theoretical models, are then used to replicate that observed behavior with specific parameters, rules, and mathematical (differential) equations. These models are then instantiated with actual parameter values (measured in experiments) and tested through computer simulations with the goal of establishing a parallel between the observed and modeled behaviors. The resulting hypotheses about insect behavior are often simple, natural and robust algorithms that may inspire distributed algorithms for various computer science problems.

A key property of many biological models is that they have a lot of built-in noise and uncertainty. In a sense, this is necessitated by the inherently noisy information that individuals have access to. Other reasons for incorporating uncertainty in these models is the possibility that individuals do not follow the rules of their algorithms precisely, and even if they do, we may not be aware of all the components of the algorithm they are running. The resulting models and algorithms may not necessarily perform correctly under every possible combination of inputs (as is usually necessary for computer algorithms); however, they are usually tolerant to perturbations of the parameters of the algorithms. Such robustness properties are not always sought after by theoretical computer scientists but they are definitely desirable in real-world distributed systems.

Combining the approaches that biologists and computer scientists use in understanding complex (distributed or biological) systems can result in (1) a more formal and mathematical understanding of the behavior of social insect colonies, and (2) new techniques to design simpler and more robust distributed algorithms. In this thesis we exploit this mutual benefit by studying three social insect colony problems (foraging, house hunting and task allocation) from a distributed computing perspective.

For each of the problems, we highlight different algorithmic characteristics like robustness properties, tolerance to noise, simplicity in terms of limited communication and computation, and insights about the actual insect behavior.

1.2 Problem Descriptions and Related Work

In this thesis we study three particular problems that are common to different species of ants (and bees) and also closely related to well-known problems in theoretical computer science. The distributed *foraging* problem refers to exploring a given area by a set of collaborative individuals in search of food or some other resource. In computer science, various such exploration problems are well-known in the plane or any other structures like graphs. The *house hunting* problem is particular to certain species of ants and bees and refers to searching and reaching consensus on a new nest for the colony to move to. House hunting is closely related to distributed consensus in simple synchronous models such as population protocols. The distributed *task allocation* problem involves a set of individuals each of which needs to choose a task (or multiple tasks) to work on with the goal of achieving a common global goal in the colony. Task allocation is also a very well-studied problem in distributed computing, usually known as resource allocation or load balancing.

The three problems described above have the advantage that they are all studied extensively by both the biology and computer science communities, so they provide opportunity to highlight tools and techniques from both areas. From a biology perspective, all three problems have computational models that suggest a possible mechanism/algorithm through which social insects solve these problems. In other words, we have some guidance in designing algorithms for solving these problems. From a distributed computing perspective, all three of these are interesting and worth studying in very simple models of complete synchrony, small number of states per agent, and very limited computation and communication capabilities. Some of these limitations of the models present difficulties in designing and analyzing algorithms but we believe these issues are inevitable in understanding real social insect colonies.

Next, we describe each of the three problems in detail and provide an overview of the relevant related work from both the computer science and biological communities. Section 1.2.1 defines the foraging problem and its relevance to searching the plane and grid exploration problems. Section 1.2.2 describes the house hunting problem together with some related consensus-like problems in various models. Section 1.2.3 defines the task allocation problem from both a biological and computer science standpoint.

1.2.1 Foraging for Food (Searching the Plane)

While foraging for food is a common problem to almost all species, social insect colonies are unique in that they use a collaborative and distributed strategy to explore an area. From a biological standpoint, the foraging problem has many variations corresponding to the specific species doing the foraging, the type of geographic areas they explore, and the type of resources they are searching for.

Here, we focus on a simple abstraction of the general foraging problem. Consider n probabilistic non-communicating agents collaboratively searching for a target in a two-dimensional grid. A target is placed within some (unknown) distance D measured in number of hops in the grid from the origin. In this setting, we study the time it takes for the first agent to reach the target. In studying solutions to the foraging problem, we consider a *selection complexity metric*, which captures the bits of memory and the range of probabilities used by a given algorithm. This combined metric is motivated by the fact that memory can be used to simulate small probability values, and small probability values can be used to approximate operations that would otherwise require more memory. More precisely, for algorithm $\mathcal{A}$, we define $\chi(\mathcal{A}) = b + \log \ell$, where b is the number of bits of memory required by the algorithm, and ℓ is the smallest value such that all probabilities used in $\mathcal{A}$ are bounded from below by $1/2^\ell$. We show that the choice of the selection metric arises naturally from the analysis of our algorithms and the lower bound.

The same abstraction of the foraging problem that we consider is also described and analyzed in recent work by Feinerman et al. [51], where it is called the *Ants Nearby Treasure Search (ANTS)* problem. The authors in [51] argue that it provides

a good approximation of insect foraging, and represents a useful intersection between biological behavior and distributed computation. The analysis in [51] focuses on the *speed-up* performance metric, which measures how the expected time to find the target improves as n increases. The authors describe and analyze search algorithms that closely approximate the straightforward $\Omega(D + D^2/n)$ lower bound[1] for finding a target placed within distance D from the origin.

Furthermore, in [51] the authors present an algorithm to find the target in optimal expected time $\mathcal{O}(D^2/n + D)$, assuming that each agent in the algorithm knows the number n of agents (but not D). For unknown n, they show that for every constant $\epsilon > 0$, there exists a uniform search algorithm that is $\mathcal{O}(\log^{1+\epsilon} n)$-competitive, but there is no uniform search algorithm that is $\mathcal{O}(\log n)$-competitive. In [50], Feinerman et al. provide multiple lower bounds on the advice size (number of bits of information the ants are given prior to the search), which can be used to store the value n, some approximation of it, or any other information. In particular, they show that in order for an algorithm to be $\mathcal{O}(\log^{1-\epsilon} n)$-competitive, the ants need advice size of $\Omega(\log \log n)$ bits. Note that this result also implies a lower bound of $\Omega(\log \log n)$ bits on the total size of the memory of the ants, but only under the condition that close-to-optimal speed-up is required. Our lower bound is stronger in that we show that there is an exponential gap of $D^{1-o(1)}$ for the maximum speed-up (with a sub-exponential number of agents). Similarly, the algorithms in [51] need $\Omega(\log D)$ bits of memory, resulting in selection metric value $\chi = \Omega(\log D)$, as contrasted with our algorithm that ensures $\chi = \mathcal{O}(\log \log D)$.

Searching and exploration of various types of graphs by single and multiple agents are widely studied in the computer science literature. Several works study the case of a single agent exploring directed graphs [3, 14, 34], undirected graphs [88, 97], or trees [5, 35]. Out of these, the following papers have restrictions on the memory used in the search: [5] uses $\mathcal{O}(\log n)$ bits to explore an n-node tree, [14] studies the power of a pebble placed on a vertex so that the vertex can later be identified, [35]

[1]The best the agents can do is split searching all the D^2 grid cells evenly among themselves; in cases where n is relatively large with respect to D^2, it still takes at least D steps for some agent to reach the target.

shows that $\Omega(\log \log n)$ bits of memory are needed to explore some n-node trees, and [97] presents a log-space algorithm for s-t-connectivity. There have been works on graph exploration with multiple agents [4, 47, 54]; while [4] and [54] do not include any memory bounds, [47] presents an optimal algorithm for searching in a grid with constant memory and constant-sized messages in a model, introduced in [48], of very limited computation and communication capabilities. This result is later extended to a model with constant memory and loneliness detection as the only communication mechanism [84]. It should be noted that even though these models restrict the agents' memory to very few bits, the fact that the models allow communication makes it possible to simulate larger memory.

So far, in the above papers, we have seen that the metrics typically considered by computer scientists in graph search algorithms are mostly the amount of memory used and the running time. In contrast, biologists look at a much wider range of models and metrics, more closely related to the physical capabilities of the agents. For example, in [7] the focus is on the capabilities of foragers to learn about different new environments, [58] considers the physical fitness of agents and the abundance and quality of the food sources, [67] measures the efficiency of foraging in terms of the energy over time spent per agent, and [99] explores the use of different chemicals used by ants to communicate with one another.

1.2.2 House Hunting (Consensus)

House hunting is the process through which some species of ants (and bees) choose a new nest for the colony to move to. The general house hunting process has (1) a search component, in which ants collectively search for and evaluate candidate nests, (2) a decision component, in which the ants distributively decide on a single nest among all candidate nests, and (3) a transportation component, in which all ants move to the chosen nest. In our abstraction of house hunting, we focus on the second component, which is inherently close to the distributed consensus problem. Next, we give some biological background on the house hunting process as performed by the *Temnothorax* ants.

Temnothorax ants live in fragile rock crevices that are frequently destroyed. It is crucial for colony survival to quickly find and move to a new nest after their home is compromised. This process is highly distributed and involves several stages of searching for nests, assessing nest candidates, recruiting other ants to do the same, and finally, transporting the full colony to the new home.

In the search phase, some ants begin searching their surroundings for possible new nests. Experimentally, this phase has not been studied much; it has been assumed that ants encounter candidate nests fairly quickly through random walking. In the assessment phase, each ant that arrives at a new nest evaluates it based on various criteria, e.g., whether the nest interior is dark and therefore likely free of holes, and whether the entrance to the nest is small enough to be easily defended. These criteria may have different priorities [65, 104] and, in general, it is assumed that nest assessments by an individual ant are not always precise or rational [102]. After some time spent assessing different nests, going back to the old nest and searching for new nests, an ant becomes sufficiently satisfied with some nest and moves on to the recruitment phase, which consists of *tandem runs* – one ant leading another ant from the old to a new nest. The recruited ant learns the candidate nest location and can assess the nest itself and begin performing tandem runs if the nest is acceptable.

At this point many nest sites may have ants recruiting to them, so a decision has to be made in favor of one nest. The ants must solve the classic distributed computing problem of consensus. One strategy that ants are believed to use is a *quorum threshold* [92, 94] – a threshold of the number of ants in a nest, that, when exceeded, indicates that the nest should be chosen as the new home. Each time an ant returns to the new nest, it evaluates (not necessarily accurately) whether a quorum has been reached. If so, it begins the transport phase – picking up and carrying other ants from the old to the new nest. These transports are generally faster than tandem runs and they conclude the house-hunting process by bringing the rest of the colony to the new nest.

We use the biological insights from various experiments to design an abstract mathematical model of the house hunting process. The model is based on a synchronous model of execution with n probabilistic ants and communication limited to

one ant leading another ant (tandem run or transport), chosen randomly from the ants at the home nest, to a candidate nest. Ants can search for new nests by choosing randomly among all k candidate nests. We do not model the time for an ant to find a nest or to lead a tandem run; each of these actions are assumed to take one round.

From a distributed computing perspective, house-hunting is closely related to the fundamental and well-studied problem of consensus [53, 75]. This makes the problem conceptually different from other ant colony inspired problems studied by computer scientists. Task allocation and foraging are both intrinsically related to parallel optimization. The main goal is to divide work optimally amongst a large number of ants in a process similar to load balancing. This is commonly achieved using random allocation or *negative feedback* [9] against work that has already been completed. In contrast, the house-hunting problem is a decision problem in which all ants must converge to the same choice. Both in nature and in our proposed algorithms, this is achieved through *positive feedback* [9], by reinforcing viable nest candidates until a single choice remains.

House hunting is a well-known problem in evolutionary biology, but the corresponding consensus problem is not very popular in theoretical computer science. However, the type of algorithms we present and their analysis is very similar to population protocols, and in particular, consensus algorithms in population protocols.

Population protocols are simple models of random interactions among agents with limited memory and computation capabilities [6, 8]. The standard model of communication in population protocols involves uniformly random interactions between pairs of computing agents through which the agents can sense each other's state. This simple exchange of a small amount of information is similar, although more general, to tandem runs in the house hunting problem. The consensus and plurality consensus problems, which resemble house hunting most closely, have been studies extensively in population protocols and similar gossip models [11, 12, 13, 36]. All of these results are set in models quite different from house hunting; population protocols tend to optimize the number of states per agent, and house hunting assumes extra capabilities of the agents like counting the number of other agents in the same nest (of the same

opinion). However, there are similarities in the tools and techniques used to analyze the correctness and performance of both types of algorithms. For both population protocols and house hunting algorithms it is useful to assume (or derive) an initial gap between the number of agents of a given opinion [11, 12]. The running times of the resulting algorithms are also similar in the two models; for n agents and k nests/opinions/colors, optimal consensus is reached in time approximately polylogarithmic in n and polynomial in k [13, 57].

1.2.3 Task Allocation (Resource Allocation)

Task allocation is the mechanism through which social insect colonies achieve division of labor, and computer systems allocate resources to various computing jobs. The goal is to assign a task to each individual by ensuring each job gets the appropriate number of workers. The challenge is to achieve this goal in a distributed way, without any central control.

The specific abstraction of the task allocation problem that we study involves a distributed process of allocating all ants to the tasks with the goal of satisfying the demand for each task. The demand of each task can be thought of as a work-rate required to keep the task satisfied. Furthermore, we assume that the demand of each task can change due to changes in the environment, so we need to repeat the distributed re-allocation process between any two such changes in demands. Since we consider all ants to be equal in skill level and preferences, the demand for each task corresponds to a certain minimum number of ants working at the task at any given time. Between any two changes in demands, each ant repeatedly decides what task to work on based on simple feedback from the environment informing the ant of the state of the task; for example, an ant may learn from the environment only whether a task needs more work or not, or, additionally, it may learn approximately how much extra work is needed to satisfy the demand of the task. We are interested in understanding how different environment feedback models affect the solvability and efficiency of the task allocation process, and how the efficiency depends on factors such as the total amount of work needed and the number of extra ants.

In particular, we consider environment feedback that provides each ant with only local information about tasks: each ant learns from the environment (1) whether it is successful at its current task, and (2) what new task it can work on. We define specific services that provide this information to each ant, and we study the resulting task allocation process from a distributed computing perspective. We show that, for all versions of the environment feedback we consider, task allocation is successful in re-assigning ants to tasks in a way that satisfies the demands of the tasks. We also analyze the time for this process terminate successfully in the presence or absence of extra ants in the colony. In particular, we focus on upper bounds of this time expressed in terms of the colony size, the number of tasks, and the total amount of new work induced by the change in demands. Our conclusions suggest that for reasonable definitions of the environment feedback, the time until ants are successfully re-allocated depends only logarithmically on the amount of work needed and decreases either linearly or logarithmically as the size of the colony increases.

Biologists have proposed multiple mechanisms that try to explain the structure of task allocation in ant colonies. Empirical work suggests that each ant chooses among the task types (brood care, foraging, nest maintenance, defense [40, 100]) based on: age [100], body size [112], genetic background [68], spatial position [107], experience [96], social interaction [61], or the need for particular work [16]. Additionally, theoretical work includes mathematical models [15, 16, 41, 60, 86, 90] that capture the essence of task allocation more abstractly. Many of these models are continuous and are based on global entities such as the rates of transition between tasks or the rates of completion of tasks, with parameters measured in experiments. Very few [86, 90] of the existing models examine factors such as group size and interaction rates between individuals, and very few [41, 60, 86] are individual-based, where group behavior is not modeled explicitly but emerges as a result of individual behavior.

Another important factor in ant task allocation is the existence of *idle* ants. Experiments have shown that a large fraction of the colony remains inactive during the task allocation process. Some hypotheses for this phenomenon include: (1) inactive ants may be spending time on non-observable activities like digesting food or dis-

tributing information throughout the colony [85, 108], and (2) inactive workers are *reserve* for times of extra needs of the colony [49, 82]. Many of these hypotheses fail to fully explain the behavior of idle ants or lack empirical evidence.

From a computer science perspective, task allocation has been studied extensively in different models of distributed computation. These results range from theoretical results that study the communication complexity of the problem [45] to practical distributed task allocation algorithms for different applications like multi-robot systems [31] and social networks [33]. A notable type of distributed task allocation problem is the *Do-All* problem [56]: a number of computing processes must cooperatively perform a number of tasks in the presence of adversity. The adversity can range from failures of processes, to asynchrony in the system, to malicious adversaries, and process behavior deviating from the specifications. Solutions to the Do-All problems and related problems have been shown to have applications is distributed search [74], distributed simulation [32], and multi-agent collaboration [2].

1.3 Results and Contributions

In this section, we summarize the main results on each one of the three problems. For the different problems, we prove slightly different types of results and consider different metrics in order to give examples of a variety of different approaches. Foraging results focus on the selection complexity metric and how it affects the time for non-communicating agents to search the plane. House hunting results focus on noise and uncertainty tolerance of algorithms and the probabilistic guarantees that allow for such tolerance. Task allocation results focus on developing insight about actual insect behavior: how the size of the colony and the number of extra ants affect the time for ants to re-allocate to tasks.

Each of these different aspects of the three problems is applicable to the other two problems as well. For example, we can study the time for ants to complete house hunting or task allocation in terms of the selection metric that we considered for foraging. It is also definitely an interesting question to study foraging in the

presence of uncertainty, or to extract some biological insights from our house hunting algorithms. In the following section, we present each of these approaches in the context of a single problem, and then we propose a number of open questions on how to apply these approaches to other problems.

1.3.1 Foraging

In our foraging work [76, 77], we generalize the problem of [51] by now also considering the selection complexity metric $\chi = b + \log \ell$ (where b is an upper bound on the number of bits and $1/2^\ell$ is a lower bound on the probability values that the algorithm can use). We begin by studying lower bounds. We identify $\log \log D$, for a target at distance within D from the origin, as a crucial threshold for the χ metric when studying the achievable speed-up [2] in the foraging problem. In more detail, our lower bound shows that for any algorithm $\mathcal{A}$ such that $\chi(\mathcal{A}) \leq \log \log D - \omega(1)$, there is a placement of the target within distance D such that the probability that $\mathcal{A}$ finds the target in less than $D^{2-o(1)}$ moves per agent is polynomially small in D. Since $\Omega(D^2)$ rounds are necessary for a single agent to explore the grid, our lower bound implies that the speed-up of any algorithm for exploring the grid with $\chi \leq \log \log D - \omega(1)$ is bounded from above by $\min\{n, D^{o(1)}\}$. For comparison, recall that because of the trivial $\Omega(D^2/n + D)$ lower bound, the optimal speed-up is $\min\{n, D\}$. At the core of our lower bound is a novel analysis of recurrence behavior of small Markov chains with probabilities of at least $1/2^\ell$.

Concerning upper bounds, we note that the foraging algorithms in [51] achieve near-optimal speed-up in n, but their selection complexity ($\chi(\mathcal{A})$) is higher than the $\log \log D$ threshold identified by our lower bound: these algorithms require sufficiently fine-grained probabilities and enough memory to randomly generate and store, respectively, coordinates up to distance at least D from the origin; this requires $\chi(\mathcal{A}) \geq \log D$. In this paper, we seek upper bounds that guarantee $\chi(\mathcal{A}) \approx \log \log D$, which is the minimum value for which good speed-up is possible. We consider two

[2] The *speed-up* of an algorithm is the ratio of the times required for a single agent and for n agents to explore the grid.

types of algorithms: non-uniform algorithms in D, which are allowed to use knowledge of D, and uniform algorithms in D, which have no information about D. All our algorithms are non-uniform in n; that is, the algorithms have knowledge of n.

We begin by describing and analyzing a simple algorithm that is non-uniform in D and has asymptotically optimal expected running time. The main idea of this algorithm is to walk up to certain points in the plane while counting approximately, and thus using little memory. We can show that this approximate counting strategy is sufficient for searching the plane efficiently. Our non-uniform algorithm uses $\chi = \log\log D + \mathcal{O}(1)$, which matches our lower bound result up to factor $1 + o(1)$.

The main idea of our uniform algorithm is to start with some estimate of D and keep increasing it while executing a corresponding version of our simple non-uniform algorithm for each such estimate. Similarly to the non-uniform algorithm, the uniform algorithm uses value of χ that is at most $\log\log D + \mathcal{O}(1)$. Additionally, we introduce a mechanism to control the trade-off between the algorithm's running time and number of bits it uses. With that goal, we let the algorithm take as a parameter a non-decreasing function $f(D)$ that represents the running-time overhead we are willing to accept; for a given function $f(D)$, the algorithm guarantees to run in $\mathcal{O}((D^2/n + D) \cdot f(D))$ moves per agent in expectation. We show that the resulting value of the selection metric $\chi = b + \log \ell$ is $\log\log D + \mathcal{O}(1)$, regardless of the choice of f, including $f = \Theta(1)$, in which case the algorithm matches the $\Omega(D^2/n + D)$ lower bound. We analyze in detail the resulting χ values for different choices of $f(D)$ and discuss what trade-offs can be achieved between the b and ℓ components of the selection metric. For example, we show that for $f(D) = \Theta(D^\epsilon)$, where $0 < \epsilon < 1$, if ℓ is sufficiently large, then $b = \log\log\log D + \mathcal{O}(1)$ bits are sufficient for the algorithm.

One of the main contributions of our work are (1) defining a new combined metric χ that captures the nature of the search problem more comprehensively compared to the standard metrics of time and space complexity. The χ metric combines the memory and probability-range metrics in a way that allows us to cover the entire range of trade-offs between the individual components and lets our results hold regardless of how an algorithm chooses to trade off the two components of the χ value. The

second main contribution is to establish $\chi \approx \log\log D$ as the threshold below which no algorithm searches the plane efficiently in terms of the speed-up the algorithm provides as the number of searchers n increases. Our lower bound indicates that any algorithm with a χ value less than the $\log\log D - \omega(1)$ threshold cannot search the plane significantly faster than n simple random walks, and our algorithms indicate that a χ value of $\mathcal{O}(\log\log D)$ is sufficient to get the optimal speed-up of $\Omega(n)$.

1.3.2 House Hunting

Our main results on the house hunting problem are a lower bound on the number of rounds required by any algorithm solving the house-hunting problem in the given model and two house-hunting algorithms [57].

The lower bound states that, under our model, no algorithm can solve the house-hunting problem in time sub-logarithmic in the number of ants. The main proof idea is that, in any step of an algorithm's execution, with constant probability, an ant that does not know of the location of the eventually-chosen nest remains uninformed. Therefore, with high probability, $\Omega(\log n)$ rounds are required to inform all n ants. This technique closely resembles lower bounds for rumor spreading in a complete graph [73], where here the rumor is the location of the chosen nest.

Our first algorithm solves the house-hunting problem in asymptotically optimal time. The main idea is a typical example of positive feedback: each ant leads tandem runs to some suitable nest as long as the population of ants at that nest keeps increasing; once the ants at a candidate nest notice a decrease in the population, they give up and wait to be recruited to another nest. With high probability, within $\mathcal{O}(\log n)$ rounds, this process converges to all n ants committing to a single winning nest. Unfortunately, this algorithm relies heavily on a synchronous execution and the ability to precisely count nest populations, suggesting that the algorithm is susceptible to perturbations of these parameters and most likely does not match real ant behavior.

The goal of our second algorithm is to be more natural and resilient to perturbations of the environmental parameters and ant capabilities. The algorithm uses a simple positive-feedback mechanism: in each round, an ant that has located a candi-

date nest recruits other ants to the nest with probability proportional to the candidate nest's current population. We show that, with high probability, this process converges to all n ants being committed to one of the k candidate nests within $\mathcal{O}(k^2 \log^{1.5} n)$ rounds. While this algorithm is not optimal, it exhibits a much more natural process of converging to a single nest.

Furthermore, for our second algorithm, we study various levels of noise and uncertainty and analyze how they affect the correctness and performance of the algorithm. One of the assumptions in the house hunting model is that ants can count the number of other ants at a given candidate nest. In fact this information is used extensively by our second algorithm to determine with what probability each ant should try to lead tandem runs. As part of our noise and uncertainty study, we analyze how the algorithm performs with approximate information about the number of ants at a given nest. In particular, for a nest of size x, and for arbitrary ϵ and c' such that $0 < \epsilon < 1$ and $c' > 2$, we assume that the estimated number of ants is in the range $[x(1-\epsilon), x(1+\epsilon)]$ with probability at least $(1 - 1/n^{c'})$; moreover, we assume the population estimate comes from some distribution that guarantees the estimate is correct in expectation. In this new model, we analyze the correctness and efficiency of the house hunting algorithm. Compared to the case of no uncertainty, we show that the new running time increases by a factor of $\mathcal{O}(1/(1-\epsilon)^2)$ and the probability of solving the problem within this time decreases by $1/n^{c'-2}$.

Furthermore, the above analysis of the uncertainty and noise tolerance of the algorithm is also useful in understanding how to combine the house hunting algorithm with a density estimation algorithm [83] that each ant uses as a subroutine to estimate the number of ants at each nest. We are interested in using such an estimate provided by a black-box subroutine that matches the uncertainty assumptions above. We show that based on the noise/uncertainty that the house hunting algorithm tolerates, we can directly plug in the density estimation algorithm in [83] and combining the resulting correctness and efficiency guarantees.

One of the main contributions of our house hunting work is the first abstract mathematical model that models this specific ant colony behavior. To put this model

in context, we provide simple matching lower bound and algorithm that establish $\Theta(\log n)$ as the time necessary and sufficient to solve house hunting. These simple results can serve as the basis for further theoretical research on the house hunting problem, in the context of the *Temnothorax* ants or a more abstract application. The second main contribution is an extremely natural algorithm that solves house hunting and is resilient to perturbations of the algorithm's parameters. The analysis of this algorithm also has implications on general probabilistic dynamics, like the *3-majority dynamic* [13] for solving consensus in population protocols. Finally, our noise analysis can be used as the basis of a more comprehensive study on how well randomized algorithms tolerate perturbations of the probabilities used in the algorithms.

1.3.3 Task Allocation

In our task allocation work, we present a mathematical model of the task allocation process in ant colonies that considers three different versions of environmental feedback. The environmental feedback consists of two components: (1) a *success* component that informs each ant whether it is successful at its current task, and (2) a *choice* component that provides the ant with an alternative task to work on. We require that the *success* component informs ants that they are successful only as long as the task at hand requires more workers; otherwise, all excess workers are considered unsuccessful. The *choice* component provides each ant with a new task from one of three possible distributions: (1) a uniformly random task, (2) a uniformly random *unsatisfied* task, and (3) a task with probability proportional to its deficit (the amount of additional work it requires).

For the various options for the *choice* feedback component, keeping the *success* component the same, we study the time to correctly re-allocate ants: the number of steps ants take until the demands of all tasks are satisfied (Figure 1-1).

The first row of the table represents the time until all ants are re-allocated to tasks ensuring that the demands of all tasks are met. In all three cases, we show that the time to re-allocate ants to tasks is logarithmic in the amount of work needed. For the first option of *choice*, the time is also linear in the number of tasks and inversely

	option (1)	option (2)	option (3)
satisfy all Φ work with prob. $\geq 1-\delta$	$\mathcal{O}(\lvert T\rvert(\frac{1}{c}))$ $(\ln\Phi + \ln(\frac{1}{\delta}))$	$\min\{\lvert T\rvert,$ $(\min\{1, \mathcal{O}(\frac{1}{\ln c})\}\cdot$ $(\ln\Phi + \ln(\frac{1}{\delta})))\}$	$\min\{\lvert T\rvert, \mathcal{O}(\frac{1}{c})$ $(\ln\Phi + \ln(\frac{1}{\delta}))\}$
satisfy $\Phi(1-\epsilon)$ work with prob. $\geq 1-\delta$	$\mathcal{O}(\lvert T\rvert(\frac{1}{c}))$ $(\ln(\frac{1}{\epsilon}) + \ln(\frac{1}{\delta}))$	$\min\{\lvert T\rvert,$ $(\min\{1, \mathcal{O}(\frac{1}{\ln c})\}\cdot$ $(\ln(\frac{1}{\epsilon}) + \ln(\frac{1}{\delta})))\}$	$\min\{\lvert T\rvert, \mathcal{O}(\frac{1}{c})$ $(\ln(\frac{1}{\epsilon}) + \ln(\frac{1}{\delta}))\}$
satisfy $\Phi - z$ work under uncertainty	did not analyze	did not analyze	$\min\{\lvert T\rvert, (\ln\Phi + \ln(\frac{1}{\delta}))$ $\mathcal{O}(\max\{\frac{1}{c}, \frac{1}{\ln(1/y)}\})\}$

Figure 1-1: Summary of Results. The values in the table are upper bounds on the time for workers to achieve a task allocation that fulfills the criteria in the first column, given a particular option for the *choice* feedback. The parameters in the table are: the number $|T|$ of tasks, the amount Φ of total work needed, the workers-to-work ratio c, the success probability $1-\delta$ and the fraction of work $1-\epsilon$ to be satisfied. For option (3), we also consider a variation where the *success* component may flip at most z bits of its outputs, and the *choice* component outputs tasks with probability lower-bounded by a $(1-y)$-fraction of the specified probabilities.

proportional to the ant-to-work ratio c. If the ants choose a task uniformly at random only among the unsatisfied tasks (option (2) for *choice*), then the resulting time to re-allocate is inversely proportional to $\ln c$. If the ants choose a task with probability proportional to the deficit (the work needed) of the task (option (3) for *choice*), the time to re-allocate is inversely-proportional to c.

The second row of the table represents the time until all ants are re-allocated such that at most $\epsilon \cdot \Phi$ of the work remains unsatisfied. In this case, we can see that the $\ln\Phi$ term is replaced by $\ln(1/\epsilon)$, indicating that the time is independent of the absolute amount of the total amount of work needed.

In the third row, we can see that if we introduce some uncertainty in the *success* and *choice* components, the ants can complete the work only to some extent and the time to re-allocate increases. More precisely, if *success* is allowed to make at most z "mistakes" (flip a 0 to a 1 or vice versa), and if the probabilities of *choice* are decreased by at most a factor of $(1-y)$, then the ants may leave z units of work unsatisfied and the running time increases as y approaches 1. Options (1) and (2) are affected even more extensively than option (3) in the case of uncertainty (not shown in the table):

depending on which tasks are affected by the mistakes of *success*, it is possible for tasks to lose more and more ants at each step.

One contribution of the task allocation work is an abstract distributed model of the task allocation process in ant colonies. The model is diverse in that it includes various possible types of input from the environment to each ant with the goal of matching the setting of different real ant colonies. Such a model can serve as the basis for further theoretical and biological research into understanding how ants achieve division of labor. The second main contribution of this work is the insight we get from our results that the key parameters that determine the efficiency of task allocation in ant colonies are (1) the amount of work needed, and (2) the ant-to-work ratio (as opposed to other parameters like the colony size $|A|$). Such an insight can serve as a hypothesis to be tested by biologists with real ant experiments like measuring the ant-to-work ratio in ant colonies and establishing whether it is the determining factor in the efficiency of task allocation.

1.4 Significance of the Results

The main significance of our results is structured around the two main goals of biological distributed algorithms: (1) use tools and techniques from distributed computing to gain insight into real biological behavior, and (2) learn from the models and algorithms occurring in ant colonies with the goal of designing better distributed algorithms. Here, we briefly summarize the contribution and significance of the main results, and we elaborate more on these points in Chapter 5.

1.4.1 Lessons for Theoretical Computer Scientists

The main insight of our results for theoretical computer scientists is to change the general approach to a problem: more flexible models, more expressive metrics, more lightweight algorithms, and to shift the complexity from the algorithm to its analysis. The goal is to have results more widely applicable to real systems, more relevant to biological systems, and possibly more interesting from a theoretical viewpoint.

From our foraging work, we see evidence that combined metrics, like the selection metric χ are more expressive and capture the nature of the problem better than considering the standard single metrics of time, space and message complexity, and then proving trade-offs between them. We believe this approach of combining metrics can be beneficial to other theoretical problems as well and help derive results that smoothly cover an entire range of metric values instead of proving a few fixed trade-offs between metrics.

Our house-hunting work illustrates a situation in which having matching lower bound and algorithm in a fixed model is not always all we need in order to solve a problem comprehensively. Designing simple and resilient algorithms sometimes requires us to treat models more flexibly and focus on how dependent the algorithm correctness is on the specific model assumptions, like for example, having the precise value of a parameter. Our house hunting work also introduces a new robustness property of randomized algorithms: their ability to tolerate perturbations of the probabilities used in the algorithms. We argue that this is an important property for randomized algorithms used in engineering and biological systems.

From most of the algorithms in this thesis, we can argue that having each agent execute the same simple rule in each round (as opposed to executing a complex multistage algorithm) helps not just in understanding and implementing the algorithm more easily, but also in making the algorithm more resilient to typical vulnerabilities of distributed algorithms like faults and asynchrony.

1.4.2 Lessons for Evolutionary Biologists

Finally, we believe biologists can also benefit from the general ideas of the results in this thesis. Our task allocation work establishes an example of applying a distributed computing approach in an attempt to answer a well-established question about the behavior of social insects: what determines the efficiency of task allocation, and what the role of idle ants in the colony is. We conjecture that abstract distributed models of insect behavior can help biologists form novel hypotheses about insect behavior and hopefully even verify these hypotheses by designing and performing new experiments.

Chapter 2

Foraging

In this chapter, we focus on the foraging problem in which a group of simple non-communicating agents is exploring the two-dimensional plane in search for a single target. The main results are two algorithms for exploring the plane and a lower bound on the selection complexity value necessary for searching the plane efficiently.

In Section 2.1, we present the system model, and formally define the search problem and the performance and selection metrics used to evaluate the algorithms.

In Section 2.2, we present a very simple algorithm that illustrates our main approach. The first algorithm is non-uniform in that it has knowledge of an upper bound D on the distance at which the target is located. This algorithm runs in optimal time and with an optimal value of the selection metric.

In Section 2.3, we generalize the main approach to uniform algorithms, which have no knowledge of an upper bound on the distance to the target. Our uniform algorithm repeatedly guesses the distance to the target and runs a version of the non-uniform algorithm at each such guess. We show that this algorithm also runs in optimal time and with an optimal value of the selection metric.

Finally, in Section 2.4, we present a lower bound that matches our upper bounds in terms of the selection metric χ, indicating that any algorithm with a lower selection metric value is asymptotically slow compared to the optimal $\mathcal{O}(D^2/n + D)$ running time. We conclude the chapter by discussing some assumptions and possible extensions of our work in Section 2.5.

2.1 Model

Our model is similar to the models in [50, 51]. We consider an infinite two-dimensional square grid with coordinates in $\mathbb{Z}^2$. The grid is to be explored by $n \in \mathbb{N}$ identical, non-communicating, probabilistic agents. Each agent is always located at a point on the grid. Agents can move in one of four directions, to one of the four adjacent grid points, but they have no information about their current location in the grid. Initially all agents are positioned at the origin. We also assume that an agent can return to the origin, and for the purposes of this paper, we assume this action is based on information provided by an oracle.[1] Without making this assumption, any algorithm automatically needs at least $\Omega(\log D)$ bits just to implement the capability to return home. Therefore, while it is a strong assumption, it lets us study the behavior of algorithms with selection complexity $\chi = o(\log D)$. In our setting, the agent returns on a shortest path in the grid that keeps closest to the straight line connecting the origin to its current position. Note that the return path is at most as long as the path of the agent away from the origin; therefore, since return paths increase the running time by at most a factor of two, and we are interested in asymptotic complexity, we ignore the lengths of these paths in our analysis.

Agents. Each agent is modeled as a probabilistic finite state automaton; since agents are identical, so are their automata. Each automaton is a tuple (S, s_0, δ), where S is a set of states, state $s_0 \in S$ is the unique starting state, and δ is a transition function $\delta : S \to \Pi$, where Π is a set of discrete probability distributions. Thus, δ maps each state $s \in S$ to a discrete probability distribution $\delta(s) = \pi_s$ on S, which denotes the probability of moving from state s to any other state in S.

For our lower bound in Section 2.4, it is convenient to use a Markov chain representation of each agent. Therefore, we can express each agent as a Markov chain with transition matrix P, such that for each $s_1, s_2 \in S$, $P[s_1][s_2] = \pi_{s_1}(s_2)$, and start state $s_0 \in S$.

[1] From a biological perspective, there is evidence that social insects use such a capability by navigating back to the nest based on landmarks in their environment [81].

In addition to the Markov chain that describes the evolution of an agent's state, we also need to characterize its movement on the grid. Let $M : S \to \{up, down, right, left, origin, none\}$ be a labeling function that maps each state $s \in S$ to an action the agent performs on the grid. For simplicity, we require $M(s_0) = origin$. Using this labeling function, any sequence of states $(s_i \in S)_{i\in\mathbb{N}}$ is mapped to a sequence of moves in the grid $(M(s_i))_{i\in\mathbb{N}}$ where $M(s_i) =$ none denotes no move in the grid (i.e., s_i does not contribute to the derived sequence of moves) and $M(s_i) =$ origin means that the agent returns to the origin, as described above.

Executions. An execution of an algorithm for some agent is given by a sequence of states from S, starting with state s_0, and coordinates of the associated movements on the grid derived from these states. Formally, an execution is defined as $(s_0, (x_0, y_0), s_1, (x_1, y_1), s_2, (x_2, y_2), \cdots)$, where $s_0 \in S$ is the start state, $(x_0, y_0) = (0, 0)$, and for each $i \geq 0$, applying the move $M(s_{i+1})$ to point (x_i, y_i) results in point (x_{i+1}, y_{i+1}). For example, if $M(s_{i+1}) =$ up, then $x_{i+1} = x_i$ and $y_{i+1} = y_i + 1$. For $M(s_{i+1}) =$ none, we define $x_i = x_{i+1}$ and $y_i = y_{i+1}$, and for $M(s_{i+1}) =$ origin, we define $(x_{i+1}, y_{i+1}) = (0, 0)$. In other words, we ignore the movement of the agent on the way back to the origin, as mentioned earlier in this section.

An execution of an algorithm with n agents is just an n-tuple of executions of single agents. For our analysis of the lower bound, it is useful to assume a synchronous model. So, we define a *round* of an execution to consist of one transition of each agent in its Markov chain. Note that we do not assume such synchrony for our algorithms.

So far, we have described a single execution of an algorithm with n agents. In order to consider probabilistic executions, note that the Markov chain (S, P) induces a probability distribution of executions in a natural way, by performing an independent random walk on S with transition probabilities given by P for each of the n agents.

Problem Statement. The goal is to find a target located at some vertex at distance (measured in terms of the max-norm) at most D from the origin in as few expected moves as possible. Note that measuring paths in terms of the max-norm gives us

a constant-factor approximation of the actual hop distance. We will consider both non-uniform and uniform algorithms with respect to D; that is, the agents may or may not know the value of D. Technically, in the case of non-uniform algorithms, each different value of D corresponds to a different algorithm. We define a family of non-uniform algorithms $\{\mathcal{A}_D\}_{D\in\mathbb{N}}$ where each $\mathcal{A}_D$ is an algorithm with parameter D.

It is easy to see (also shown in [51]) that the expected running time of any algorithm is $\Omega(D + D^2/n)$ even if agents know n and D and they can communicate with each other. This bound can be matched if the agents know a constant-factor approximation of n [51], but as mentioned in the introduction, the value of the selection metric χ (introduced below) in that specific algorithm is $\Omega(\log D)$. For simplicity, throughout this paper we will consider algorithms that are non-uniform in n, i.e., the agents' state machine is allowed to depend on n. One can apply a technique from [51] that the authors use to make their algorithms uniform in n, in order to generalize our results and obtain an algorithm that is uniform in both D and n, at the cost of an $\mathcal{O}(\log^{1+\epsilon} n)$-factor running time overhead.

Metrics. For the problem defined above, we consider both a performance and a selection metric and study the trade-offs between the two. We will use the term *step* of an agent interchangeably with a transition of the agent in the Markov chain. We define a *move* of the agent to be a step that the agent performs in its Markov chain resulting in a state labeled *up*, *down*, $left$, or $right$.

For our main performance metric, we focus on the asymptotic running time in terms of D and n; more precisely, we are interested in the metric M_{moves}: the minimum over all agents of the number of moves of the agent until it finds the target. Note that for this performance metric we exclude states labeled *none* and *origin* in an execution of an agent. We already argued that the *origin* states increase the running time by at most a factor of two. We consider the transitions to *none* states to be part of an agent's local computation. Intuitively, we can think of consecutive transitions to *none* states to be grouped together with the first transition to a non-*none* state and considered a single move. Both our algorithm bounds and our lower bound are

expressed in terms of M_{moves}. For the proof of our lower bound, it is also useful to define a similar metric in terms of the steps of an agent. We define the metric M_{steps} to be the minimum over all agents of the number of steps of the agent until it finds the target. This metric is used only as a helper tool in our lower bound analysis.

The selection metric of a state automaton (and thus of a corresponding algorithm) is defined as $\chi(\mathcal{A}) = b + \log \ell$, where $b = \lceil \log |S| \rceil$ is the number of bits required to encode all states from S and $1/2^\ell$ is a lower bound on $\min\{P[s, s'] \mid s, s' \in S \wedge P[s, s'] \neq 0\}$, that is, on the smallest non-zero probability value used by the algorithm. We further motivate this choice in Section 2.2 and Section 2.3, where we describe different trade-offs between the performance metric and the values of b and ℓ.

2.2 Non-uniform Algorithm

In this section we present an algorithm in which the value of D is available to the algorithm. Fix $D \in \mathbb{N}$ and define algorithm $\mathcal{A}_D \in \{\mathcal{A}_D\}_{D \in \mathbb{N}}$ based on the following general approach: each agent chooses a vertical direction (up or down) with probability 1/2, walks in that direction for a random number of steps that depends on D, then does the same for the horizontal direction, and finally returns to the origin and repeats this process. In Theorem 2.2.5, we show that the expected minimum over all agents of the number of moves of the agent to find a target at distance up to D from the origin is at most $\mathcal{O}(D^2/n + D)$.

Let coin C_p denote a coin that shows tails with probability p. Assuming coin $C_{1/D}$ is available to the algorithm, we present Algorithm 1, accompanied by a state machine representation (for simplicity of presentation the state machine does not depict the states labeled *none*). Note that the state machine is not an exact representation of the code in Algorithm 1 because the algorithm uses only coin flips while the state machine has more than two outgoing transitions per state. However, by checking the probabilities associated with each action, it is easy to verify that the behaviors of the state machine and the algorithm are identical. If we were to construct a state machine that matches the algorithm precisely, it would require four bits to represent,

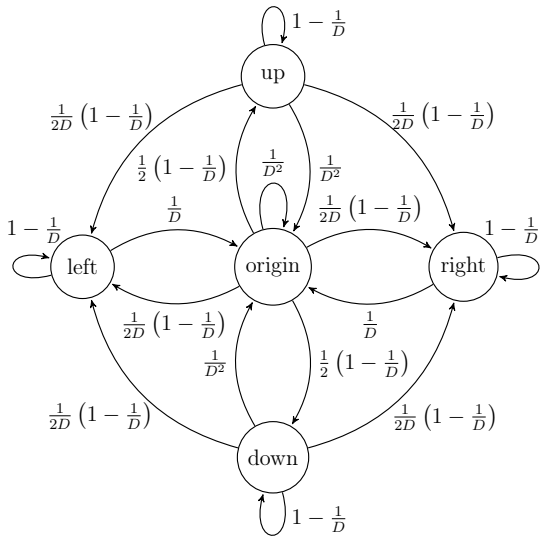

Figure 2-1: State machine representation of Algorithm 1. State names match the values of the labeling function.

as opposed to three bits in the current state machine.

Later in this section we present Algorithm $\mathcal{A}_D$, which is a slightly modified version of Algorithm 1 that removes the need for coin $C_{1/D}$. In Theorem 2.2.7 we show that Algorithm $\mathcal{A}_D$ guarantees that $\chi = \log\log D + \mathcal{O}(1)$.

Algorithm 1: Non-uniform Search Algorithm.

```
while true do
    if coin C_{1/2} shows heads then
        while coin C_{1/D} shows heads do
            move up
    else
        while coin C_{1/D} shows heads do
            move down
    if coin C_{1/2} shows heads then
        while coin C_{1/D} shows heads do
            move left
    else
        while coin C_{1/D} shows heads do
            move right
    return to the origin
```

Fix an arbitrary point (x, y) in the grid, where $x, y \in \mathbb{Z}$ and $|x|, |y| \leq D$; this point represents the location of the target. The algorithms presented in this section

are analyzed with respect to the number of moves until some agent explores grid point (x, y) and, thus, finds the target. For Lemmas 2.2.1, 2.2.2, 2.2.3, and 2.2.4 consider an arbitrary fixed agent.

Let T denote the number of moves for the agent to complete an iteration of the outer loop of the algorithm. Also, let event S (for successful) be the event that the agent finds the target in the given iteration. Similarly, let event U (for unsuccessful) denote the event that the agent does not find the target in the given iteration. Since the length and success probability of each iteration is the same, we do not index the length T of the iteration and the events S and U by the index of the iteration. Next, we bound $\mathbb{E}[T]$, $\mathbb{E}[T \mid U]$, and $\mathbb{E}[T \mid S]$.

Lemma 2.2.1. $\mathbb{E}[T] \leq 2D$.

Proof. In each iteration, the agent performs one move up or down for each consecutive toss of coin $C_{1/D}$ showing heads, and then one move right or left for each consecutive toss of coin $C_{1/D}$ showing heads. Each of these walks is D steps long in expectation, so it follows that $E[T] \leq 2D$. □

Lemma 2.2.2. $\mathbb{E}[T \mid S] \leq 2D$.

Proof. This holds because in a successful iteration the agent makes at most D horizontal moves followed by at most D vertical moves. □

Lemma 2.2.3. $\mathbb{E}[T \mid U] \leq 2\mathbb{E}[T]$.

Proof. First, we bound the probability that the agent does not find the target in a given iteration. If $y > 1$, with probability $1/2$ coin $C_{1/2}$ shows tails, so the agent does not move up, and consequently, it does not find the target in this iteration. Symmetrically, if $y < 1$, with probability $1/2$ the agent does not find the target. Overall, in a given iteration, with probability at least $1/2$, the target is not found. By the law of total expectation it follows that:

$$\mathbb{E}[T] \geq \Pr[U] \cdot \mathbb{E}[T \mid U] \geq \left(\frac{1}{2}\right) \mathbb{E}[T \mid U].$$

□

Since all iterations by all agents are identical and independent, instead of analyzing iterations performed by the n agents in parallel, we can consider an infinite sequence of consecutive iterations performed by a single agent. In the next theorem, we will assign these iterations to the n agents in a round-robin way and analyze the resulting parallel running time.

Let random variable N denote the number of unsuccessful iterations before the first successful iteration, and let the sequence $T_1, T_2, \cdots$ denote the lengths of the iterations performed by the algorithm. Since the lengths of iterations are identical, we know that for all $i \geq 1$, $\mathbb{E}[T_i] = \mathbb{E}[T]$.

Lemma 2.2.4. *$E[N] \leq 16D$.*

Proof. We bound the probability for the agent to find the target in a single iteration.

Suppose the target is located in the first quadrant. With probability $1/4$, an agent moves up and right during an iteration of the algorithm. The probability that the walk up halts after exactly x steps is $(1-1/D)^x(1/D) \geq (1-1/D)^D(1/D) \geq 1/(4D)$.

The probability that the walk right halts after $y \leq D$ steps is at least $(1 - 1/D)^D \geq 1/4$. Hence, in each iteration, an agent finds the target with probability at least $1/(16D)$. The same holds for a target located in any of the other quadrants. Therefore, $E[N] \leq 16D$. □

Theorem 2.2.5. *Let each of n agents execute Algorithm 1. For a target located within distance $D > 1$ from the origin, $\mathbb{E}[M_{moves}] \leq (64D^2)/n + 6D = \mathcal{O}(D^2/n + D)$.*

Proof. First, we assign the N unsuccessful iterations to the n agents round robin. Therefore, each agent executes a total of at most $\lceil N/n \rceil$ unsuccessful iterations. Fix the agent that executes the following iterations: $1, 1+n, 1+2n, \cdots, 1+(\lceil N/n \rceil - 1)n$ and note that no other agent executes more iterations.

Next, we bound the value of $\mathbb{E}[M_{moves}]$ by the expected duration of the unsuccessful iterations $\mathbb{E}[\sum_{i=0}^{\lceil N/n \rceil - 1} T_{i \cdot n+1}]$ of the fixed agent plus the expected duration of a successful iteration by the agent that actually finds the target. Note that this is an upper bound because it is possible that the successful agent finds the target before

the fixed agent completes its unsuccessful iterations.

$$\begin{aligned}
\mathbb{E}[M_{moves}] &\leq \mathbb{E}\left[\sum_{i=0}^{\lceil N/n\rceil-1} T_{i\cdot n+1}\right] + \mathbb{E}[T_{N+1}] \\
&= \sum_{j=0}^{\infty}\left(\mathbb{E}\left[\sum_{i=0}^{\lceil j/n\rceil-1} T_{i\cdot n+1} \mid N=j\right] + \mathbb{E}[T_{j+1} \mid N=j]\right)\cdot \Pr[N=j] \\
&= \sum_{j=0}^{\infty}\left(\sum_{i=0}^{\lceil j/n\rceil-1} \mathbb{E}[T_{i\cdot n+1} \mid N=j] + \mathbb{E}[T_{j+1} \mid N=j]\right)\cdot \Pr[N=j].
\end{aligned}$$

Since $N = j$ and $i \cdot n + 1 \leq j$, we know that $T_{i\cdot n+1}$ is an unsuccessful iteration. Therefore, $\mathbb{E}[T_{i\cdot n+1} \mid N = j] = \mathbb{E}[T \mid U]$. For the same reason, $\mathbb{E}[T_{j+1} \mid N = j] = \mathbb{E}[T \mid S]$.

$$\begin{aligned}
\mathbb{E}[M_{moves}] &\leq \sum_{j=0}^{\infty}\left(\sum_{i=0}^{\lceil j/n\rceil-1} \mathbb{E}[T \mid U] + \mathbb{E}[T \mid S]\right)\cdot \Pr[N=j] \\
&\leq \sum_{j=0}^{\infty}\left(\left(\frac{j}{n}+1\right)\mathbb{E}[T \mid U] + \mathbb{E}[T \mid S]\right)\cdot \Pr[N=j] \\
&= \mathbb{E}[T \mid U]\sum_{j=0}^{\infty}\left(\frac{j}{n}+1\right)\cdot \Pr[N=j] + \mathbb{E}[T \mid S] \\
&= \mathbb{E}[T \mid U]\cdot\left(\frac{\mathbb{E}[N]}{n}+1\right) + \mathbb{E}[T \mid S] \\
&\leq 4D\cdot\left(\frac{16D}{n}+1\right) + 2D \qquad \text{by Lemmas 2.2.2, 2.2.3, and 2.2.4} \\
&= \frac{64D^2}{n} + 6D.
\end{aligned}$$

Note, it is technically possible that $\Pr[N = \infty] \neq 0$ implying that we may need an unbounded number of iterations to find the target. However, this is not the case because it is easy to see that each iteration terminates in a finite and bounded number of rounds with probability 1, so $\Pr[N = \infty] = 0$. □

We now generalize this algorithm to one that uses probabilities lower bounded by $1/2^\ell$ for some given $\ell \geq 1$. This is achieved by the following subroutine, which

implements a coin that shows tails with probability $1/2^{k\ell}$ using a biased coin that shows tails with probability $1/2^{\ell}$, for $\ell \geq 1$.

Algorithm 2: coin(k, ℓ): Generate coin $C_{1/2^{k\ell}}$ using coin $C_{1/2^{\ell}}$.

```
for i = 0 ··· k do
    if C_{1/2^ℓ} shows heads then
        return heads
return tails
```

Lemma 2.2.6. *Algorithm 2 returns tails with probability $1/2^{k\ell}$ and uses $\lceil \log k \rceil$ bits of memory.*

Proof. From the code it follows that the action on the second line is performed only if none of the outcomes of the coin flips are tails. Since each coin shows tails with probability $1/2^{\ell}$ and there is a total if k coin flips, the probability of all of them being tails is $1/2^{k\ell}$. Since the entire state of the algorithm is the loop counter, it can be implemented using $\lceil \log k \rceil$ bits of memory. □

Next, we show how to combine Algorithm 1 and Algorithm 2, and we analyze the performance and selection complexity of the resulting algorithm. Given a biased coin $C_{1/2^{\ell}}$, we construct algorithm $\mathcal{A}_D$ by replacing the lines where coin $C_{1/D}$ is tossed in Algorithm 1 with a copy of Algorithm 2, with parameters $k = \lceil \log D/\ell \rceil$ and ℓ.

Theorem 2.2.7. *Let each of n agents run algorithm $\mathcal{A}_D$. For a target located within distance $D > 1$ from the origin, $\mathbb{E}[M_{moves}] = \mathcal{O}(D^2/n + D)$. Moreover, algorithm $\mathcal{A}_D$ satisfies $\chi(\mathcal{A}_D) = \log\log D + \mathcal{O}(1)$.*

Proof. By Lemma 2.2.6, Algorithm 2 run with parameters $k = \lceil (\log D)/\ell \rceil$ and $\ell' = (\log D)/k \leq \ell$, generates coin flips with probability $1/D$ of showing tails. Also, Algorithm 2 does not generate any moves of the agents on the grid. So, the correctness and time complexity of the algorithm follow from Theorem 2.2.5. Finally, by Lemma 2.2.6 and the fact that Algorithm 1 uses only 3 bits, it follows that:

$$\begin{aligned}\chi(\mathcal{A}_D) &= b + \log \ell = \log\lceil \log D/\ell \rceil + \log \ell + 3 \\ &\leq \log\log D - \log \ell + 1 + \log \ell + 3 = \log\log D + \mathcal{O}(1).\end{aligned}$$

□

2.3 Uniform Algorithm

In this section, we generalize the results from Section 2.2 to derive an algorithm that is uniform in D. The main difference is that now each agent maintains an estimate of D that it increases until it finds the target. For each estimate, an agent simply executes a subroutine similar to algorithm $\mathcal{A}_D$. Moreover, the algorithm in this section takes as a parameter a non-decreasing function $f : \mathbb{Z}^+ \to [1, \infty)$ and ensures that the resulting running time $\mathbb{E}[M_{moves}]$[2] is $\mathcal{O}((D^2/n + D) \cdot f(D))$. In other words, given a desired (asymptotic) approximation ratio to the optimal value of $\Theta(D^2/n + D)$, we provide an algorithm that solves the problem in the required expected time and we calculate the necessary value of χ for such a solution. The analysis of the value of $\mathbb{E}[M_{moves}]$ is presented in a general way and works for any function f such that $f(2) \geq 128 \ln 8$. For the analysis of the resulting value of the selection metric χ and the trade-off between its components, we plug in different values of f.

We show that for any sufficiently large function f, the selection metric achieved by the algorithm is $\chi = \mathcal{O}(\log \log D)$. We also consider specific functions f. For example, we consider $f(x) = \Theta(1)$ and we conclude that in this case the algorithm uses $b = \mathcal{O}(\log \log D)$ bits, regardless of the value of ℓ. For $f(x) = \Theta(x^\epsilon)$, where $0 < \epsilon < 1$, however, we show that if $\ell = \log D - \log \log D$, then $b = \mathcal{O}(\log \log \log D)$ bits are sufficient for the algorithm. At the end of the section we also discuss other options for the function f and additional considerations for the approximation factor.

The rest of this section is organized as follows: Section 2.3.1 defines a useful sequence of estimates of D using the function f, Sections 2.3.2 and 2.3.3 present the algorithm and running time analysis, respectively, and Section 2.3.4 includes the selection metric analysis for the algorithm.

[2]Note that fixing a uniform algorithm, a distance $D \in \mathbb{N}$ and a target location within distance D from the origin is sufficient to define a probability distribution over all executions of the algorithm with respect to the given target location. The metric M_{moves} and its expectation are defined over that distribution.

2.3.1 Definition and Properties of T_i and D_i

We construct two infinite sequences, a sequence $\mathcal{T} = (T_1, T_2, \cdots)$ of non-negative reals, and a sequence $\mathcal{D} = (D_1, D_2, \cdots)$ of non-negative integers. Here, D_i represents the i'th estimate of D and T_i represents a bound on the expected time an agent spends searching for the target within distance D_i (including the overhead in the running time defined by f) in order to find a target within this distance with sufficiently large probability. Such a table of values can be pre-calculated for a given choice of f and then utilized by the algorithm. For a given function f, the sequences $\mathcal{D}$ and $\mathcal{T}$ will be hardwired into the agents' automaton, so that the only values the agent has to store in its main memory are the current index i and the specific values of D_i and T_i corresponding to that index; however, the agent never needs to store the entire sequences of values. Recall that our definition of b depends only on the number of states of the agents' automata. Thus, it represents the number of "read-write" memory bits required to record an agent's state. The sequences T_i and D_i are fixed and thus can be stored in "read-only" memory. For simplicity, we assume an agent can compute these values online for simple enough choices of f (without violating the memory and probability restrictions). A detailed analysis and discussion of the memory used by the algorithm are presented in Section 2.3.4.

We define the following set of constraints on the values of the $\mathcal{D}$ and $\mathcal{T}$ sequences:

$$D_0 = 2 \tag{2.1}$$

$$\text{For each } i \in \mathbb{N},\ D_i > 0 \tag{2.2}$$

$$\text{For each } i \in \mathbb{N},\ i \geq 1,\ T_i = \frac{D_{i-1}^2}{n} \cdot f(D_{i-1}) \tag{2.3}$$

$$\text{For each } i \in \mathbb{N},\ i \geq 1,\ T_{i+1} = \frac{T_i}{4} \cdot e^{\frac{f(D_{i-1})}{32} \cdot \frac{D_{i-1}^2}{D_i^2}} \tag{2.4}$$

Before we proceed to the algorithm, we show that these constraints uniquely define the sequences $\mathcal{T}$ and $\mathcal{D}$, and then we prove that these sequences are strictly increasing. For the results below, recall that we assume that f is non-decreasing and that $f(2) \geq 128 \ln 8$.

Lemma 2.3.1. *Fix n, D_{i-1} and T_i, for any $i \in \mathbb{N}$. Then, Equations (2.3) and (2.4) have a unique solution for D_i and T_{i+1}.*

Proof. We need to show that given D_{i-1} and T_i we can calculate D_i and T_{i+1}. Based on the two defining equations for T_{i+1}, it suffices to show that the equation below always has a unique solution:

$$e^{\frac{f(D_{i-1})}{32} \cdot \frac{D_{i-1}^2}{D_i^2}} \cdot \frac{D_{i-1}^2}{4n} \cdot f(D_{i-1}) - \frac{D_i^2}{n} \cdot f(D_i) = 0.$$

Note that the left hand side is a continuous function (assuming we extend the domain to the reals) and $D_{i-1} > 0$ is already fixed. Moreover, the left hand side is of the form $ae^{b/D_i^2} - cD_i^2 f(D_i)$ for positive a, b, and c that are independent of D_i. Since f is non-decreasing, $f(D_i)$ can be uniformly bounded from above when considering $D_i \to 0$ (e.g. by $f(D_{i-1})$). The left hand side remains positive, so it is bounded from below by $ae^{b/D_i^2} - c'D_i^2$ for positive a, b, c' if $D_i <= D_{i-1}$.

For $D_i \to 0$, the left hand side tends to ∞, whereas for $D_i \to \infty$, it tends to $-\infty$. Hence, by the mean value theorem, there is always a solution D_i to the above equation. Moreover, the left hand side is strictly decreasing in D_i (for $D_i > 0$), implying that this solution is unique. From the solution for D_i we can then easily compute the value of T_{i+1}. □

Lemma 2.3.2. *For each $i \in \mathbb{N}$, $i \geq 1$, $T_{i+1} \geq 2T_i$.*

Proof. Fix some $i \in \mathbb{N}$ and consider two cases based on the values of D_i and D_{i-1}. Also, recall that $D_0 \geq 2$.

Case 1: $D_i \geq 2D_{i-1}$. By Equation (2.3) and the fact that f is non-decreasing, we have:

$$\frac{T_{i+1}}{T_i} = \frac{D_i^2 \cdot f(D_i)}{D_{i-1}^2 \cdot f(D_{i-1})} \geq \frac{(2D_{i-1})^2}{D_{i-1}^2} > 2.$$

Case 2: $D_i < 2D_{i-1}$. By Equation (2.4) and the fact that $f(2) \geq 128 \ln 8$:

$$T_{i+1} = e^{\frac{f(D_{i-1})}{32} \cdot \frac{D_{i-1}^2}{D_i^2}} \cdot \frac{T_i}{4} \geq e^{\frac{f(2)}{32} \cdot \frac{1}{4}} \cdot \frac{T_i}{4} \geq 2T_i.$$

□

Note that, based on Lemma 2.3.2 and the assumption that f is a non-decreasing function, it follows from Equation (2.3) that $\mathcal{D}$ is a strictly increasing sequence.

Before using the sequences $\mathcal{D}$ and $\mathcal{T}$ in the uniform search algorithm, we give an example of these sequences for the very simple case when $f = \Theta(1)$ (in particular, we consider $f = 80$ and $n = 100$). Each D_i in the sequence below represents a (rounded-up) guess of D, and the corresponding T_i represents the (rounded-up) expected number of rounds the algorithm spends searching at distance D_i.

$\mathcal{D}$: (2, 2.4, 2.9, 3.4, 4.1, 4.9, 5.9, 7.1, 8.4, 10.1, $\cdots$)

$\mathcal{T}$: ($\perp$, 3.2, 4.6, 6.6, 9.4, 13.5, 19.3, 27.6, 39.6, 56.7, $\cdots$)

2.3.2 Subroutines and Algorithm

To simplify the presentation, we break up the main algorithm into two subroutines. The first subroutine is similar to Algorithm 1; however, instead of using the actual distance D as a parameter, the following algorithm uses an estimate D_i of D.

Algorithm 3: search(i): Visit each point of a square of side length D_i with probability at least $1/(16D_i)$.

```
if if C_{1/2} shows heads then
    while C_{1/D_i} = heads do
        move up
else
    while C_{1/D_i} = heads do
        move down
if C_{1/2} shows heads then
    while C_{1/D_i} = heads do
        move right
else
    while C_{1/D_i} = heads do
        move left
```

Lemma 2.3.3. *For a fixed $i \in \mathbb{N}$ and each point (x, y) where $x, y \in \mathbb{Z}$ and $|x|, |y| \leq D_i$, if Algorithm 3 is called at the origin, then it visits point (x, y) with probability at least $1/(16D_i)$.*

Proof. The proof is identical to the proof of Lemma 2.2.4 (D is replaced by D_i). □

Next, in Algorithm 4, we use Algorithm 3 to efficiently search an infinite grid using n agents. Intuitively, the algorithm iterates through different values of the outer-loop parameter i, which correspond to the different estimates of D, increasing according to the sequence $\mathcal{D}$. For each such estimate, the algorithm executes a number of calls to the search subroutine with parameter i. However, since agents have limited memory and limited probability values, we can only count the number of such calls to the search routine approximately. We do so by repeatedly tossing a biased coin and calling the search algorithm as long as the coin shows heads.

Algorithm 4: Uniform Search Algorithm.

```
for i = 1, ··· do
    Let x = T_i/D_i
    while C_{1/x} = heads do
        search(i)
        return to the origin
```

2.3.3 Running Time Analysis of the Uniform Algorithm

For the rest of this section, fix some $D \in \mathbb{N}$ and a point (x, y) in the grid, where $x, y \in \mathbb{Z}$ and $|x|, |y| \leq D$, that represents the location of the target. Having fixed Algorithm 4, distance D, and a location for the target within distance D from the origin, the metric M_{moves} (and its expectation) can now be defined with respect to the distribution of executions of Algorithm 4.

Throughout the following proofs, we refer to an iteration of the outermost loop as a phase and we refer to a call to search(i) as an iteration. In Lemma 2.3.4, we calculate the expected number of moves for an agent to complete phase i. In Lemma 2.3.8, we calculate the probability that some agent finds the target in some phase

i. Finally, we use these intermediate results to prove the main result of this section, Theorem 2.3.9, which shows that the expected number of moves for the first agent to find a target within distance D from the origin is $\mathcal{O}((D + D^2/n) \cdot f(D))$. The structure of the proof is very similar to that of the non-uniform algorithm in Section 2.2; however, here we consider phases instead of iterations.

For Lemmas 2.3.4, 2.3.5, 2.3.6, 2.3.7, and 2.3.8 fix an arbitrary agent. Denote by R_i the number of moves for the agent to complete phase i.

Lemma 2.3.4. $\mathbb{E}[R_i] \leq 2T_i$.

Proof. We bound R_i by summing over all possible numbers of iterations in phase i (indexed by j) and, inside that sum, summing over the possible lengths of each such iteration (indexed by k). For this analysis, we can assume that an iteration is always finished by the agent executing it, even if the agent happens to find the target in the middle of the iteration. The factor of 2 before the second sum is due to the fact that each iteration consists of a vertical and a horizontal set of moves.

$$\begin{aligned}
\mathbb{E}[R_i] &= \sum_{j=0}^{\infty} \frac{D_i}{T_i} \cdot \left(1 - \frac{D_i}{T_i}\right)^j \cdot j \cdot 2 \sum_{k=0}^{\infty} \frac{1}{D_i} \left(1 - \frac{1}{D_i}\right)^k \cdot k \\
&= \sum_{j=0}^{\infty} \frac{D_i}{T_i} \cdot \left(1 - \frac{D_i}{T_i}\right)^j \cdot j \cdot 2(D_i - 1) \\
&= \left(\frac{T_i}{D_i} - 1\right) 2\,(D_i - 1) \leq \frac{T_i}{D_i} \cdot 2D_i = 2T_i.
\end{aligned}$$

□

Let N_i denote the number of iterations in phase i (until the target is found or until the end of the phase).

Lemma 2.3.5. $\mathbb{E}[N_i] \leq T_i/D_i$.

Proof. By the pseudocode:

$$\mathbb{E}[N_i] = \sum_{j=0}^{\infty} \frac{D_i}{T_i} \cdot \left(1 - \frac{D_i}{T_i}\right)^j \cdot j = \frac{T_i}{D_i} \left(1 - \frac{D_i}{T_i}\right) \leq \frac{T_i}{D_i}.$$

□

Let event S_i (for successful) be the event that the agent finds the target in phase i. Similarly, let event U_i (for unsuccessful) denote the event that the agent does not find the target in phase i. Next, we bound $\mathbb{E}[R_i \mid U_i]$ and $\mathbb{E}[R_i \mid S_i]$.

Lemma 2.3.6. $\mathbb{E}[R_i \mid U_i] \leq 4T_i$.

Proof. For each $k \geq 1$, let X_k denote the length of the k'th iteration of phase i. Since the lengths of all iterations in a given phase identically distributed, let $\mathbb{E}[X]$ denote their common expected length. Finally, let U denote the event that a given iteration is unsuccessful. Reasoning identically to Lemma 2.2.3, $\mathbb{E}[X \mid U] \leq 4D_i$.

$$\begin{aligned}
\mathbb{E}[R_i \mid U_i] &= \mathbb{E}\left[\sum_{k=0}^{N_i-1} X_{k+1} \mid U_i\right] \\
&= \sum_{j=0}^{\infty}\sum_{k=0}^{j-1} \mathbb{E}[X_{k+1} \mid U] \cdot \Pr[N_i = j \mid U_i] \\
&= \sum_{j=0}^{\infty}\sum_{k=0}^{j-1} \mathbb{E}[X \mid U] \cdot \Pr[N_i = j \mid U_i] \\
&= \sum_{j=0}^{\infty} j \cdot \mathbb{E}[X \mid U] \cdot \Pr[N_i = j \mid U_i] \\
&= \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i \mid U_i].
\end{aligned}$$

It remains to show that $\mathbb{E}[N_i \mid U_i] \leq \mathbb{E}[N_i]$. Consider the two probabilities: $\Pr[N_i = j]$ and $\Pr[N_i = j \mid U_i]$. There are two possible ways that $N_i = j$: either the phase ends because the agent finds the target, or the phase ends because the coin $C_{1/x}$ shows tails. On the other hand, conditioning on U_i, the only way $N_i = j$ is if the coin $C_{1/x}$ shows tails. Therefore, $\Pr[N_i = j] \geq \Pr[N_i = j \mid U_i]$, and so $\mathbb{E}[N_i \mid U_i] \leq \mathbb{E}[N_i]$.

By Lemma 2.3.5, $\mathbb{E}[R_i \mid U_i] = \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i \mid U_i] \leq \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i] \leq 4T_i$. □

Lemma 2.3.7. $\mathbb{E}[R_i \mid S_i] \leq 4T_i + 2D$.

Proof. For each $k \geq 1$, let X_k denote the length of the k'th iteration of phase i. Since all iterations in a given phase are identical in length, let $\mathbb{E}[X]$ denote their common

expected length. Finally, let U denote the event that a given iteration is unsuccessful. Reasoning identically to Lemma 2.2.3, $\mathbb{E}[X \mid U] \leq 4D_i$. Note that each successful iteration is of length at most $2D$.

We can bound $\mathbb{E}[R_i \mid S_i]$ by summing over all $N_i - 1$ unsuccessful iterations in phase i and then adding a successful iteration of length at most $2D$.

$$\mathbb{E}[R_i \mid S_i] \leq \mathbb{E}\left[\sum_{k=0}^{N_i-2} X_{k+1} \mid S_i\right] + 2D.$$

Reasoning similarly to Lemma 2.3.6, we get $\mathbb{E}[R_i \mid S_i] \leq \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i - 1 \mid S_i] + 2D$. Again, we just need to show that $\mathbb{E}[N_i \mid S_i] \leq \mathbb{E}[N_i]$, and the same reasoning as in Lemma 2.3.6 holds.

By Lemma 2.3.5, we have $\mathbb{E}[R_i \mid S_i] \leq \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i \mid S_i] + 2D \leq \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i] + 2D \leq 4T_i + 2D$. □

Let i_0 be the minimum phase such that $D \leq D_{i_0}$.

Lemma 2.3.8. *For each phase $i \geq i_0$, the probability that an agent finds the target in phase i is at least $1 - 2e^{-T_i/(32D_i^2)}$.*

Proof. By Lemma 2.3.5, the expected number of iterations in phase i is at least T_i/D_i.

Fix the number of coin flips performed by the agent in phase i. Since the coin flips are independent, we can apply a Chernoff bound (Theorem A.1.4 in the Appendix), showing that the probability that fewer than $T_i/(2D_i)$ searches are executed in total is at most $e^{-T_i/(12D_i)}$.

Condition on the event that there are at least $T_i/(2D_i)$ iterations in phase i. We can apply Lemma 2.3.3 to bound the probability of finding the target in phase i because $D_i \geq D_{i_0} \geq D$, so the probability to miss the target in all iterations of phase i is at most:

$$\left(1 - \frac{1}{16D_i}\right)^{T_i/(2D_i)} \leq e^{-T_i/(32D_i^2)}.$$

Therefore, the probability that the agent finds the target in phase i is at least:

$$\left(1 - e^{-T_i/(12D_i)}\right)\left(1 - e^{-T_i/(32D_i^2)}\right) \geq 1 - 2e^{-T_i/(32D_i^2)}.$$

□

Let random variable N denote the number of unsuccessful phases $i \geq i_0$ before the first successful phase.

Theorem 2.3.9. *Let each of n agents execute Algorithm 4. For a target located within distance $D > 1$ from the origin, $\mathbb{E}[M_{moves}] = 20(D^2/n + 2D) \cdot f(D) = \mathcal{O}(D^2/n + D) \cdot f(D)$.*

Proof. Before we proceed to bound $\mathbb{E}[M_{moves}]$, we calculate a few terms that will appear in the bound of $\mathbb{E}[M_{moves}]$. Let $p_i = 2e^{-(f(D_{i-1})/32)(D_{i-1}^2/D_i^2)}$.

By Lemma 2.3.8, the probability that none of the agents find the target in phase i is at most

$$2e^{-nT_i/(32D_i^2)} \leq 2e^{-(f(D_{i-1})/32)(D_{i-1}^2/D_i^2)} = p_i. \tag{2.5}$$

For $j \geq 0$, $\Pr[N = j]$ is at most the probability that none of the n agents finds the target in the j phases following phase i_0. So:

$$\Pr[N = j] \leq \prod_{i=i_0}^{i_0+j-1} p_i. \tag{2.6}$$

Next, note that the value of p_i appears in the definition of T_i in Equation (2.4). So, we can write $T_{i+1} = T_i/(2p_i)$. Also, for any $j \geq 0$:

$$T_{i_0+j} = T_{i_0} \prod_{i=i_0}^{i_0+j-1} \frac{1}{2p_i}.$$

Finally, note that for any $j, k \geq 1$:

$$\sum_{i=j}^{k} T_i \leq T_k \sum_{i=j}^{k} 2^{j-k} \leq 2T_k. \tag{2.7}$$

We can bound M_{moves} by summing over the first i_0 phases, the next N unsuccessful phases, and the last successful phase. We assume an arbitrary fixed agent executes each one of these phases. Note that although this fixed agent is not guaranteed to be the one that finds the target, the expected number of moves of the agent that finds the target is bounded by the expected number of moves of the fixed agent.

Recall that by Lemmas 2.3.4, 2.3.6, and 2.3.7, we have $\mathbb{E}[R_i] \leq 2T_i$, $\mathbb{E}[R_i \mid U_i] \leq 4T_i$, and $\mathbb{E}[R_i \mid S_i] \leq 4T_i + 2D$, respectively.

$$\begin{aligned}
&\mathbb{E}[M_{moves}] \\
&\leq \sum_{i=1}^{i_0-1} \mathbb{E}[R_i] + \mathbb{E}\left[\sum_{i=i_0}^{i_0+N-1} R_i\right] + \mathbb{E}[R_{i_0+N}] \\
&\leq \sum_{i=1}^{i_0-1} 2T_i + \sum_{j=0}^{\infty} \sum_{i=i_0}^{i_0+j-1} \mathbb{E}[R_i \mid N=j] \cdot \Pr[N=j] + \sum_{j=0}^{\infty} \mathbb{E}[R_{i_0+j} \mid N=j] \cdot \Pr[N=j] \\
&\leq \sum_{i=1}^{i_0-1} 2T_i + \sum_{j=0}^{\infty} \sum_{i=i_0}^{i_0+j-1} \mathbb{E}[R_i \mid U_i] \cdot \Pr[N=j] + \sum_{j=0}^{\infty} \mathbb{E}[R_{i_0+j} \mid S_i] \cdot \Pr[N=j] \\
&\leq \sum_{i=1}^{i_0-1} 2T_i + \sum_{j=0}^{\infty} \sum_{i=i_0}^{i_0+j-1} 4T_i \cdot \Pr[N=j] + \sum_{j=0}^{\infty} (4T_{i_0+j} + 2D) \cdot \Pr[N=j] \\
&\leq \sum_{i=1}^{i_0-1} 2T_i + \sum_{j=0}^{\infty} \sum_{i=i_0}^{i_0+j} 4T_i \cdot \Pr[N=j] + 2D \\
&\leq 4T_{i_0} + \sum_{j=0}^{\infty} 8T_{i_0+j} \cdot \Pr[N=j] + 2D && \text{by Equation (2.7)} \\
&\leq 4T_{i_0} + 8T_{i_0} \sum_{j=0}^{\infty} \prod_{i=i_0}^{i_0+j-1} \left(\frac{1}{2p_i}\right) \prod_{i=i_0}^{i_0+j-1} p_i + 2D && \text{by Equations (2.5) and (2.6)} \\
&\leq 4T_{i_0} + 8T_{i_0} \sum_{j=0}^{\infty} 2^{-j} + 2D \leq 4T_{i_0} + 16T_{i_0} + 2D \\
&\leq 20\left(\frac{D_{i_0-1}^2}{n}\right) f(D_{i_0-1}) + 2D \leq 20\left(\frac{D^2}{n} + 2D\right) f(D) && \text{by Equation (2.3).}
\end{aligned}$$

As a technical note, if the right hand side of Equation (2.6) does not go to 0 as j goes to infinity, then we cannot use the method above to bound $\mathbb{E}[M_{moves}]$ because $\Pr[N = \infty] \neq 0$ implying that we may need an unbounded number of phases to find the target. However, this is not the case because it is easy to see that each phase terminates in a finite and bounded number of rounds with probability 1, so $\Pr[N = \infty] = 0$. □

2.3.4 Selection Metric Analysis

In this section, we analyze the selection metric requirements of Algorithm 4. First, in Section 2.3.4, we prove some bounds on the χ selection metric for an arbitrary function f (subject to the constraints listed at the beginning of Section 2.3) used to define the sequences $\mathcal{D}$ and $\mathcal{T}$. Next, in Sections 2.3.4 and 2.3.4, we substitute some specific functions for f in order to get closed-form results for some different values of χ. For the memory component, b, of the selection metric, we consider only dynamically-changing memory (variables that take on different values throughout the execution of the algorithm); for example, the loop counter i and the corresponding value T_i/D_i in Algorithm 4 are dynamically changing, while the entire pre-computed sequences of T_i's and D_i's are not dynamically changing because the algorithm uses them only as a look-up table and does not modify these sequences.

General Analysis

The memory requirements of the algorithm can be split into three parts: (1) bits to represent the counter value i, (2) bits to implement the search routine with argument i, and (3) bits to implement coin $C_{1/x}$ for $x = T_i/D_i$. Similarly to Section 2.3.3, let i_0 be the minimum phase such that $D \leq D_{i_0}$.

Note that, technically, these memory requirements are random variables because the algorithm does not guarantee that once it reaches phase i_0 it finds the target with probability 1. In other words, it is possible for the algorithm to reach phases greater than i_0 before actually finding the target. Therefore, for the purposes of this

analysis, we will assume that after the algorithm reaches phase i_0 it may run out of memory, and if it does so, it just stops incrementing the phases. Since it has already reached the right distance to search, by the analysis in Section 2.3.3, we know that this restriction does not prevent the algorithm from finding the target.

Part (1) depends on how fast the sequence of D_i's grow, which depends on our choice of function f. Since i_0 is the maximum phase index that the algorithm can reach before it finds the target, we need to account for the bits used to represent i_0.

For part (2), we can use the subroutine in Algorithm 2 to implement the specific coin values we need using only the coin we have, $C_{1/2^\ell}$ (recall that ℓ is a parameter that determines the smallest probability the algorithm may use). By Lemma 2.2.6, it follows that we need at most $\log\log D_{i_0} - \log \ell$ bits to implement coin $C_{1/D_{i_0}}$. The remaining part of the search subroutine uses only a constant number of bits.

For part (3), we calculate the number of bits to implement coin $C_{1/x}$ for $x = T_i/D_i$:

$$\frac{T_i}{D_i} = \frac{D_{i-1}^2 \cdot f(D_{i-1})}{n \cdot D_i} \leq D_{i-1} \cdot f(D_{i-1}) \leq D_{i_0-1} \cdot f(D_{i_0-1}).$$

The exact number of bits used to implement coin $C_{1/(D_{i_0-1} \cdot f(D_{i_0-1}))}$ depends on the choice of f.

In the following two subsections, we analyze two choices for the function f that will let us calculate specific values for the selection metric χ and the relationship between the b and ℓ components. Namely, we consider f: $f(D_i) = c$, for some constant $c \geq 128$ and $f(D_i) = \Theta(D_i^\epsilon)$ for some $0 < \epsilon < 1$. Our analysis shows that in the first case, the algorithm uses $\chi = 2\log\log D + \mathcal{O}(1)$, which we also show to be the case in general, for any (larger than some constant) function f. In the second case, the same value of χ is sufficient; however, we show the additional property that if ℓ is large enough, then $b = \log\log\log D + \mathcal{O}(1)$, while $\chi = b + \log \ell = \mathcal{O}(\log\log D)$ is still satisfied.

$f(D_i) = c$ for some constant $c \geq 128$

Theorem 2.3.10. *For $f(D_i) = c$, $c \geq 128$, Algorithm 4 uses $\chi = 2\log\log D + \mathcal{O}(1)$.*

Proof. Substituting f in Equations (2.3) and (2.4), we get the following equations:

$$\begin{aligned} T_i &= 128 \cdot \frac{D_{i-1}^2}{n} \\ T_{i+1} &= 128 \cdot \frac{D_i^2}{n} \\ T_{i+1} &= \frac{T_i}{4} \cdot e^{\frac{3D_{i-1}^2}{D_i^2}} \end{aligned}$$

Substituting the value of T_i in the above equations, we get:

$$\begin{aligned} \frac{128D_i^2}{n} &= \frac{128D_{i-1}^2}{4n} \cdot e^{\frac{3D_{i-1}^2}{D_i^2}} \\ \ln\left(\frac{D_i^2}{D_{i-1}^2}\right) &= \frac{3D_{i-1}^2}{D_i^2} - \ln 4 \end{aligned}$$

Note that if $D_i/D_{i-1} \leq 2$, then we get that $\ln(D_i^2/D_{i-1}^2) \leq \ln 4$, and so:

$$\begin{aligned} \frac{D_{i-1}^2}{D_i^2} &\leq \frac{2\ln 4}{3} \\ \frac{D_i}{D_{i-1}} &\geq \sqrt{\frac{3}{2\ln 4}}. \end{aligned}$$

We are guaranteed that $D_i/D_{i-1} \geq \sqrt{3/(2\ln 4)} > 1$. Since $D_0 = 2$, it is easy to see that each D_i is at least $2(\sqrt{3/(2\ln 4)})^{i-1}$. Therefore, $i_0 - 2 = \mathcal{O}(\log D_{i_0-1}) = \mathcal{O}(\log D)$ and $\log\log D + \mathcal{O}(1)$ bits are sufficient to represent index i_0.

Additionally, as mentioned above, the algorithm uses at most $\log\log D_{i_0} - \log \ell = \log\log D + \mathcal{O}(1) - \log \ell$ bits to implement coin $C_{1/D_{i_0}}$. Similarly, the algorithm uses $\log \mathcal{O}(\log D_{i_0-1})) - \log \ell = \log\log D + \mathcal{O}(1) - \log \ell$ bits to implement coin $C_{1/(D_{i_0-1} \cdot f(D_{i_0-1}))}$. Since we implement each coin only after we are done with the previous one, the same bits can be reused to toss both coins.

The resulting value of the selection metric is $\chi = b + \log \ell = 2\log\log D + \mathcal{O}(1)$. □

Note that, for this choice of f, the value of b used by the algorithm is $\Theta(\log\log D)$ in the worst case, regardless of the value of ℓ. This is true because larger values of ℓ can help implement the coins used in the algorithm with fewer bits, but it does not

affect the fact that $\Theta(\log\log D)$ bits are used to represent index i_0. So, even if large values of ℓ are available to the algorithm, the memory requirement remains the same.

Next, we generalize the result of Theorem 2.3.10 to any choice of the function f that is polynomial in D.

Corollary 2.3.11. *Algorithm 4 uses $\chi = 2\log\log D + \mathcal{O}(1)$, for any function $f(D_i) \geq 128$ and $f(D_i) \leq T(D_i)$ for any polynomial T.*

Proof. By Theorem 2.3.10, Algorithm 4 uses $\chi = 2\log\log D + \mathcal{O}(1)$, for $f(D_i) = c$, where $c \geq 128$. For any faster-growing function, the values of subsequent D_i's grow faster, and consequently, the target value D_{i_0} is reached faster. Therefore, for any function growing faster than a constant, i_0 can be represented with fewer bits, compared to the case of $f(D_i) = c$. Since $f(D_i)$ is at most polynomial in D_i, the number of bits sufficient to implement coins $C_{1/D_{i_0}}$ and $C_{1/(D_{i_0}\cdot f(D_{i_0-1}))}$ is asymptotically the same as in the case of $f(D_i) = c$. Therefore, for any $f(D_i) \geq 128$ that is at most polynomial in D_i, Algorithm 4 uses $\chi = 2\log\log D + \mathcal{O}(1)$. □

$f(D_i) = \Theta(D^\epsilon)$ for some $0 < \epsilon < 1$.

Next, we consider $f(D_i) = \Theta(D^\epsilon)$ and we would like to show that for this choice of f, unlike the case of $f = \Theta(1)$, the algorithm uses fewer bits of memory, provided that the value of ℓ is sufficiently large. In particular, we show that if $\ell = \log D - \log\log D$, then $b = \log\log\log D + \mathcal{O}(1)$. Note that these values of b and ℓ still satisfy the selection metric value of $\chi = b + \log\ell = \mathcal{O}(\log\log D)$.

Theorem 2.3.12. *For $f(D_i) = \Theta(D_i^\epsilon)$, where $0 < \epsilon < 1$, and $\ell = \log D - \log\log D$, Algorithm 4 uses $b = \log\log\log D + \mathcal{O}(1)$.*

Proof. Substituting f in Equations (2.3) and (2.4), we get the following equations:

$$
\begin{aligned}
T_i &= \frac{D_{i-1}^2}{n} \cdot \Theta(D_{i-1}^\epsilon) \\
T_{i+1} &= \frac{D_i^2}{n} \cdot \Theta(D_i^\epsilon) \\
T_{i+1} &= \frac{T_i}{4} \cdot e^{\frac{\Theta(D_{i-1}^\epsilon)}{128}\frac{D_{i-1}^2}{D_i^2}}
\end{aligned}
$$

Substituting the value of T_i in the above equations, we get:

$$\begin{aligned}
\frac{D_i^2}{n} \cdot \Theta(D_i^\epsilon) &= \frac{D_{i-1}^2}{n} \cdot \Theta(D_{i-1}^\epsilon) \cdot e^{\frac{\Theta(D_{i-1}^\epsilon)}{128} \frac{D_{i-1}^2}{D_i^2}} \\
\frac{\Theta(D_i^{2+\epsilon})}{\Theta(D_{i-1}^{2+\epsilon})} &= e^{\frac{\Theta(D_{i-1}^{2+\epsilon})}{D_i^2}} \\
\Theta(\ln D_i) - \Theta(\ln D_{i-1}) &= \frac{\Theta(D_{i-1}^{2+\epsilon})}{D_i^2} \\
D_i^{2+o(1)} &\geq D_{i-1}^{2+\epsilon} \\
D_i &\geq D_{i-1}^{1+\Omega(\epsilon)}
\end{aligned}$$

Given that $D_0 = 2$, it is easy to see that each D_i is at least $\Theta(2^{(1+\Omega(\epsilon))^{i-1}})$. Therefore, $i_0 - 2 = \mathcal{O}(\log\log D_{i_0-1}) = \mathcal{O}(\log\log D)$, so the algorithm uses at most $\log\log\log D + \mathcal{O}(1)$ bits to represent index i_0. □

Discussion

First, note that some other functions, not considered above, lie asymptotically between $\Theta(1)$ and $\Theta(D_i^\epsilon)$. For example, two such functions are $\Theta(\log D_i)$ and $2^{\Theta(\log^\lambda D_i)}$ where $\lambda \in [0, 1]$. We can perform similar calculations to those in Sections 2.3.4 and 2.3.4, to show that, for both of these functions, the algorithm uses $\mathcal{O}(\log\log D)$ bits to encode the index i_0. In other words, we get the same asymptotic bounds for χ as in the case of $f(D_i) = \Theta(1)$. In contrast, $f(D_i) = \Theta(D^\epsilon)$ is the slowest-growing function we could identify for which (given a large enough value of ℓ) the memory used by the algorithm is $\mathcal{O}(\log\log\log D)$ bits, substantially smaller that $\mathcal{O}(\log\log D)$. This implies that as the desired approximation to the running time, specified by f, grows to $\Theta(D^\epsilon)$ or higher (e.g. polynomial in D), the memory used by the algorithm decreases to $\mathcal{O}(\log\log\log D)$ bits. Functions larger that polynomial in D are not of particular interest here because for the resulting running time there are much simpler ways to search the plane, for example, simple random walks.

The selection metric bounds in this section can be shown to be generalizations of the one we get in [76] where the approximation factor of the running time is

$2^{\mathcal{O}(\ell)}$. Performing similar calculations to those in Section 2.3.4, we can see that in the case of $f(D_i) = 2^{\mathcal{O}(\ell)}$, the estimates of D grow as $D_i/D_{i-1} \geq 2^{\mathcal{O}(\ell)}$. Therefore, we need a selection metric value of $\chi = \mathcal{O}(\log \log D)$ for the algorithm to satisfy this approximation factor, which is asymptotically the same as in [76].

2.4 Lower bound

In this section, we present a lower bound on the number of rounds necessary for an algorithm to find a target placed within distance D from the origin, with non-negligible probability, if the algorithm satisfies $\chi(\mathcal{A}) \leq \log \log D - \omega(1)$. The rest of this section is structured as follows: in Section 2.4.1, we state the main theorem with respect to the metric M_{steps} and non-uniform algorithms and give an overview of the proof, in Section 2.4.2, we present the proof in detail, and, finally, in Sections 2.4.3 and 2.4.4, we extend the lower bound to the case of the metric M_{moves} for non-uniform and uniform algorithms, respectively.

2.4.1 Theorem for M_{steps} and non-uniform algorithms

Fix a family of non-uniform algorithms $\{\mathcal{A}_D\}_{D\in\mathbb{N}}$ and fix some constant $c > 1$. Let $f_1, f_2 : \mathbb{N} \to [1, \infty)$ be arbitrary functions such that $f_1(D) = \omega(1)$, $f_2(D) = o(1)$, and $f_2(D) = \omega(1/2^{f_1(D)} + \log \log D)/\log D)$. Also, let T be an arbitrary polynomial and fix a constant c_n such that $T(D) \leq D^{c_n}$ for any D.

Theorem 2.4.1. *For each $D \in \mathbb{N}$, $D > 1$ and each $n \in \mathbb{N}$, $n \leq T(D)$, assume algorithm $\mathcal{A}_D$ with n agents satisfies $\chi(\mathcal{A}_D) = b + \log \ell \leq \log \log D - f_1(D)$. Then, there exists a placement (x, y), $|x|, |y| \leq D$ of the target, such that, with probability at least $1 - 1/D^c$, algorithm $\mathcal{A}_D$ satisfies $M_{steps} > D^{2-f_2(D)}$ for this placement (x, y).*

Proof Overview: Here we provide a high-level overview of our main proof argument. We fix an algorithm $\mathcal{A}_D$ for $D \in \mathbb{N}$, $D > 1$, and focus on executions of this algorithm of length $D^{2-o(1)}$ rounds. We prove that since agents have $o(\log D)$ states, they "forget" about past events too fast to behave substantially differently from a

biased random walk. Note that a random walk is essentially memoryless since each new step is independent of the previous steps, so it cannot "remember" what it has visited already.

More concretely, first we show in Corollary 2.4.3 that after $D^{o(1)}$ initial rounds each agent is located in some recurrent class C of the Markov chain. We use this corollary to prove, in Corollary 2.4.4, that after the initial $D^{o(1)}$ rounds each agent either does not return to the origin, or it keeps returning every $D^{o(1)}$ rounds, so it does not explore much of the grid. Therefore, throughout the rest of the proof we can ignore the states labeled "origin".

Assume (for the purposes of this overview) there is a unique stationary distribution of C.[3] Since there are few states and non-zero transition probabilities are bounded from below, standard results on Markov chains imply that taking $D^{o(1)}$ steps from any state in the recurrent class will result in a distribution on the states of the class that is (almost) indistinguishable from the stationary distribution (Corollary 2.4.6); in other words, any information agents try to preserve in their state will be lost quickly with respect to D.

The next step in the proof is a coupling argument. We split up the rounds in the execution into groups such that within each group, rounds are sufficiently far apart from one another for the above "forgetting" to take place. For each group, we show that drawing states independently from the stationary distribution introduces only a negligible error (Lemma 2.4.7 and Corollary 2.4.8). Doing so, we can apply a Chernoff bound to each group, yielding that an agent will not deviate substantially from the expected path it takes when, in each round, it draws a state according to the stationary distribution and executes the corresponding move on the grid (Lemma 2.4.10 and Corollary 2.4.12). Taking a union bound over all groups, it follows that, with high probability, each agent will not deviate from a straight line (the expected path associated with the recurrent class it ends up in) by more than distance $o(D/|S|)$, where S is the number of states of the Markov chain. It is crucial here that the

[3]This holds only if the induced Markov chain on the recurrent class is aperiodic, but the reasoning is essentially the same for the general case. We handle this technicality at the beginning of Section 2.4.2.

corresponding region in the grid, restricted to distance D from the origin, has size $o(D^2/|S|)$ and depends only on the component of the Markov chain the agent ends up in. Therefore, since there are no more than $|S|$ components, taking a union bound over all agents shows that with high probability together they visit an area of $o(D^2)$.

2.4.2 Proof

Fix some $D \in \mathbb{N}$, $D > 1$; this also fixes an algorithm $\mathcal{A}_D \in \{\mathcal{A}_D\}_{D\in\mathbb{N}}$. Assume $\chi(\mathcal{A}_D) = b + \log \ell \leq \log\log D - f_1(D)$. Also, fix constants $c' \geq c + c_n + 5$ and $d > 2(c' + 2)$.

We define the following parameters that depend on algorithm $\mathcal{A}_D$ (and its Markov chain representation) and will be used throughout the rest of this section.

- Let p_0 denote the smallest non-zero probability in the Markov chain describing $\mathcal{A}_D$. By assumption, $p_0 \geq 1/2^\ell$.
- Let b denote the number of bits required to represent the Markov chain describing $\mathcal{A}_D$. By assumption, $2^b \geq |S|$, where S is the set of states in the Markov chain.
- Let $R_0 = c'|S|p_0^{-|S|} \ln D = D^{o(1)}$, and let $\beta = 2d|S|^2 p_0^{-2|S|^2} \ln D = D^{o(1)}$. These parameters will be used to denote "chunks" of rounds by the end of which we show that the Markov chain reaches some well-behaved states.
- Let $\Delta = D^{2-f_2(D)}$. The values of the functions f_1 and f_2 are chosen carefully in order to ensure that $\Delta = o(D^2/(\beta|S|^2 \log D))$. This parameter will be used to represent the total running time of the algorithm of choice.

Consider the probability distribution of executions of $\mathcal{A}_D$ of length $R_0+\Delta$ rounds. We break the proof down into three main parts. Sections 2.4.2 and 2.4.2 use standard Markov chain techniques to derive some results for our constrained (in terms of number of states and range of probabilities) Markov chain, and Section 2.4.2 applies these results to the movement of the agents in the grid. First, in Section 2.4.2, we show

that, with high probability, after a certain number of initial rounds each agent is in a recurrent class of its Markov chain. Until we resume the proof of Theorem 2.4.1, we also condition on this recurrent class not containing any states labeled *origin*. Next, in Section 2.4.2, we show that if we break down the execution into sufficiently large blocks of rounds, then we can assume that, with high probability, the steps associated with rounds in different blocks do not depend on each other. Finally, in Section 2.4.2, we focus on the movement of the agents in the grid, derived from these "almost" independent steps, and we show that with high probability, among all points at distance $\mathcal{O}(D)$ from the origin, the agents will only explore a total area of $o(D^2)$.

Initial steps in the Markov chain

In this subsection we prove some properties of the states of the Markov chain of each agent after some number of initial rounds. Let random variable $C(r)$ denote the recurrent class of the Markov chain in which an agent is located immediately after r rounds; if the agent is in a transient state immediately after r rounds, then $C(r) = \perp$.

First, we show that for any state s of the Markov chain, if state s is always reachable, then, with high probability, the agent visits state s within $D^{o(1)}$ rounds.

Lemma 2.4.2. *Let s be an arbitrary state. Then, with probability at least $1 - 1/D^{c'}$, one of the following is true: (1) the agent visits state s within R_0 rounds, or (2) the agent is located in some state s' immediately after $r \leq R_0$ rounds such that s is not reachable from s'.*

Proof. We will prove by induction on $i \in \mathbb{Z}^+$ that, with probability at least $1 - (1 - p_0^{|S|})^i$, one of the following is true: (1) the agent visits state s within $|S|i$ rounds, or (2) the agent is located in some state s' immediately after $r \leq |S|i$ rounds such that s is not reachable from s'.

In the base case, for $i = 1$, if state s is not reachable from the initial state, then part (2) holds; otherwise, the probability that state s is reached within $|S|$ rounds is at least $p_0^{|S|}$. For the inductive hypothesis assume that with probability at least $1 - (1 - p_0^{|S|})^i$, one of the following is true: (1) the agent visits state s within $|S|i$

rounds, or (2) the agent is located in some state s' immediately after $r \leq |S|i$ rounds such that s is not reachable from s'. Following the same argument as in the base case, if state s is no longer reachable, then part (2) holds; otherwise, with probability at least $p_0^{|S|}$, state s is reached within $|S|$ rounds. Overall, with probability at least $1-(1-p_0^{|S|})^{i+1}$, one of the following is true: (1) the agent visits state s within $|S|(i+1)$ rounds, or (2) the agent is located in some state s' immediately after $r \leq |S|(i+1)$ rounds such that s is not reachable from s'.

Evaluating this probability for $i = R_0/|S|$, we get:

$$1 - \left(1 - p_0^{|S|}\right)^{R_0/|S|} = 1 - \left(1 - p_0^{|S|}\right)^{p_0^{-|S|}c' \ln D} \geq 1 - e^{-c' \ln D} = 1 - \frac{1}{D^{c'}}.$$

Therefore, with probability at least $1 - 1/D^{c'}$, one of the following is true: (1) the agent visits state s within R_0 rounds, or (2) the agent is located in some state s' immediately after $r \leq R_0$ rounds such that s is not reachable from s'. □

In the following corollary we show that within R_0 rounds, with high probability, an agent is located in some recurrent class of the Markov chain.

Corollary 2.4.3. *With probability at least $1 - 1/D^{c'}$, it is true that $C(R_0) \neq \perp$.*

Proof. First, we derive a Markov chain from the original Markov chain as follows. We identify all recurrent states in the original Markov chain and we merge them all into a single recurrent state s_C of the derived Markov chain (see Figure 2-2).

By definition of a recurrent class and because there is only one such class, for each state s in the derived Markov chain, the recurrent state s_C is always reachable from s. By Lemma 2.4.2, with probability at least $1 - 1/D^{c'}$, the agent visits s_C within R_0 rounds. This implies that in the original Markov chain, with probability at least $1 - 1/D^{c'}$, the agent visits *some* recurrent state $s \in C(R_0)$, such that $C(R_0) \neq \perp$, within R_0 rounds. □

In Corollary 2.4.3, we showed that, with high probability, within R_0 rounds the agent is located in some recurrent class $C(R_0) \neq \perp$. Since the agent does not leave that class in subsequent rounds, we will refer to it by C (a random variable). Finally,

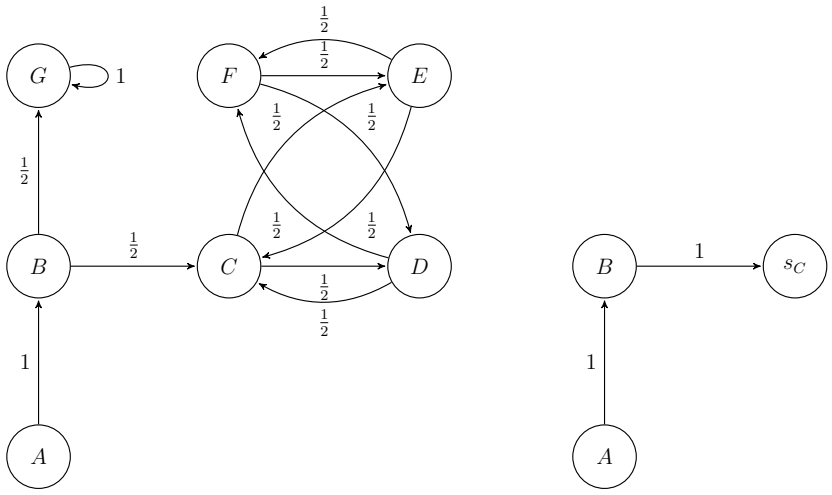

Figure 2-2: On the left: simple example of Markov chain with start state A. The recurrent classes are $\{G\}$ and $\{C, D, E, F\}$. On the right: all recurrent states merged into a single state s_C.

we show that, with high probability, either recurrent class C does not contain any states labeled *origin*, or the agent keeps returning to the origin often.

Corollary 2.4.4. *With probability at least $1 - 1/D^{c'-3}$, at least one of the following is true: (1) for all rounds r, where $R_0 \leq r \leq \Delta + R_0$, the agent visits a state labeled origin at least once between rounds r and $r + R_0$, or (2) none of the states in C are labeled origin.*

Proof. Consider any fixed execution prefix of length R_0 rounds and condition on the event that the agent is in some state s in recurrent class C at the end of the prefix. If C contains no states labeled *origin*, then (2) holds.

Otherwise, each state $s' \in C$ labeled *origin* is reachable from state s in each round $r \geq R_0$. By Lemma 2.4.2, with probability at least $1 - 1/D^{c'}$, the agent visits state s' within R_0 rounds. Since the agent does not leave C, we can repeat this argument for each group of R_0 rounds in the execution. In an execution of length $R_0 + \Delta$ rounds, there are $o(D^2)$ groups of R_0 rounds. By a union bound, with probability at least $1 - 1/D^{c'-2}$, for all rounds r, where $R_0 \leq r \leq \Delta + R_0$, the agent visits a state labeled origin at least once between rounds r and $r + R_0$.

By the law of total probability, since all execution prefixes of R_0 rounds are dis-

joint, the conclusion above holds for all executions. Combining this result and Corollary 2.4.3 by a union bound shows that, with probability at least $1 - 1/D^{c'-3}$, at least one of the two statements of the corollary holds. □

Until we resume the proof of Theorem 2.4.1, we consider executions after round R_0 and condition on the event that the agent is in some recurrent class C which does not contain any states labeled *origin*. In the proof of Theorem 2.4.1 at the end of this section, we refer to Corollary 2.4.4 in order to incorporate the probability of this event into the final probability bound. For convenience, we refer to the remaining Δ rounds of the execution as round numbers 1 to Δ. This numbering is used throughout Sections 2.4.2 and 2.4.2; at the end of Section 2.4, when we resume the proof of Theorem 2.4.1, we incorporate the initial rounds to conclude the final result about the entire execution.

Moves drawn from the stationary distribution

Fix an arbitrary recurrent class C of the Markov chain. Let t denote the period of the Markov chain (an aperiodic chain has period $t = 1$). We apply Theorem A.2.1 in the Appendix to C and denote by $G_1, \cdots, G_t$ the equivalence classes based on the period t whose existence is guaranteed by the theorem.

Consider blocks of rounds of size $\beta = 2d|S|^2 p_0^{-2|S|^2} \ln D = D^{o(1)}$. We assume that β is a multiple of t. Otherwise, we can use $t\lceil \beta/t \rceil = \mathcal{O}(\beta)$ assuming $t \in \mathcal{O}(\beta)$; this is true because $t \leq |S|$ and $p_0^{-2|S|^2} \ln D \geq 1$ because of the restriction on χ. We define groups of rounds such that each group contains one round from each block. Formally, for $1 \leq i \leq \beta$ and $j \in \mathbb{N}_0$, group B_i contains round numbers $i + j\beta \leq \Delta$. Observe that, based on this definition, immediately after each round from a given group, the agent is in some state from the same class $G \subseteq C$ that is recurrent and closed under P^t, where P is the probability matrix of the original Markov chain. By [52][Chapter XV.7], there is a unique stationary distribution π of the Markov chain on G induced by P^t (see Figure 2-3 for an illustration of P^t and the equivalence classes $G_1, \cdots, G_t$).

The following lemma bounds the value of $\pi(s')$ for each state $s' \in G$ in the Markov

chain on G induced by P^t.

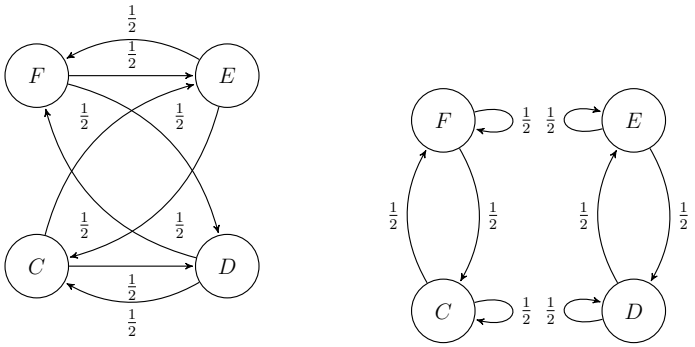

Figure 2-3: On the left: a recurrent class, with transition matrix P and period 2, of the Markov chain from Figure 2-2. On the right: Markov chain induced by P^2. The equivalence classes here are $\{C, F\}$ and $\{D, E\}$.

Lemma 2.4.5. *Assume $|G| > 1$. Then, for each $s' \in G$, and each constant $c'' \geq 2^{-f_1(D)}$:*

$$\frac{1}{D^{c''}} \leq \pi(s') \leq 1 - \frac{1}{D^{c''}}$$

Proof. Since any state $s' \in G \subseteq C$ is reachable from any state $s'' \in G \subseteq C$ by a sequence of at most $|C| - 1 < |S|$ state transitions, then it follows that, for each $s' \in G$:

$$\pi(s') = \sum_{s'' \in G} P^t(s'', s')\pi(s'') \geq p_0^{|S|} \sum_{s'' \in G} \pi(s'') = p_0^{|S|}$$

Since $|G| > 1$, this implies that $\pi(s') \leq 1 - p_0^{|S|}$ for each $s' \in G$.

Finally, we use the assumption $b + \log \ell \leq \log \log D - f_1(D)$ in order to bound $p_0^{|S|}$:

$$p_0^{|S|} \geq \left(\frac{1}{2^\ell}\right)^{2^b} \geq 2^{-\ell 2^b} \geq 2^{-2^{\log \ell + b}} \geq 2^{-2^{\log \log D - f_1(D)}} = D^{-2^{-f_1(D)}} \geq \frac{1}{D^{c''}}.$$

□

We say that two discrete probability distributions π_1 and π_2, with the same do-

main, are *D-approximately equivalent* iff $\|\pi_1 - \pi_2\| \leq 1/D^d$ where $\|\cdot\|$ denotes the ∞-norm on the given space.

Let π_s denote the probability distribution on G of the possible states of the agent immediately after round $r + \beta$, conditioned on the agent being in state $s \in G$ immediately after r rounds.

Next, we show that the distribution π_s and the stationary distribution π of the Markov chain are D-approximately equivalent. We obtain the following corollary of Lemma 2 from [101] (also stated as Lemma A.2.2 in the Appendix) applied to the Markov chain induced by the matrix P^t restricted to class G.

Corollary 2.4.6. *For each state $s \in G$, π_s and π are D-approximately equivalent.*

Proof. We can apply Lemma A.2.2 in order to show that the stationary distribution π and the actual distribution π_s are very close to each other (D-approximately equivalent).

Since β is a multiple of t, we can consider the probability matrix P^t, which by Theorem A.2.1 induces a Markov chain on G. Also, since the Markov chain induced by P^{tk_0} is aperiodic and since C is a recurrent class, we can apply Lemma A.2.3. It essentially states that there exists an integer r such that there is a walk of length exactly r between any pair of states in the Markov chain. Moreover, we are guaranteed that $r \leq 2|S|^2$.

Next, we apply Lemma A.2.2 to this chain with the following parameters: $k_0 = r/t$, $Q(s) = 1$ (i.e., $Q(s') = 0$ for all $s' \in G \setminus \{s\}$), and $k = \beta/t$. We also need that $P^{tk_0}(s', s) \geq \epsilon$ for each $s' \in G$ and a suitable $\epsilon > 0$. We can choose $\epsilon = p_0^{2|S|^2}$ guaranteeing that $P^{tk_0}(s', s) = P^r(s', s) \geq p_0^r \geq p_0^{2|S|^2} = \epsilon$.

By Lemma A.2.2, it follows that:

$$\|\pi_s - \pi\| \leq (1-\epsilon)^{\lfloor k/k_0 \rfloor} = \left(1 - p_0^{2|S|^2}\right)^{dp_0^{-2|S|^2} \ln D} \leq e^{-d \ln D} = \frac{1}{D^d}.$$

Therefore, distributions π_s and π are D-approximately equivalent. □

Having established that the distribution of states in the Markov chain is very close

to the stationary distribution of the Markov chain, in the next lemma, we quantify this difference by introducing a new distribution π' that denotes the "gap" between the actual distribution and the stationary distribution of the Markov chain.

Lemma 2.4.7. *Let $1 \leq i \leq \beta$ and $\tau = i \bmod t$ for some integer τ. Then, for each state $s \in G$ there exists a probability distribution π'_s such that:*

$$\forall r \in B_i, r \leq \Delta - \beta : \ \frac{1}{D^{c'+2}} \ \pi'_s + \left(1 - \frac{1}{D^{c'+2}}\right) \pi = \pi_s$$

Proof. If $G = \{s\}$, then, trivially, $\pi(s) = \pi_s(s) = 1$ and we choose $\pi'_s(s) = 1$. For the rest of the proof, assume that $|G| > 1$. We use the equation in the statement of the lemma to define π'_s:

$$\forall s' \in G : \ \pi'_s(s') = D^{c'+2} \left(\pi_s(s') - \left(1 - \frac{1}{D^{c'+2}}\right) \pi(s')\right).$$

We need to show that π'_s is indeed a probability distribution; that is, we show that the sum of $\pi'_s(s')$ for all states $s' \in G$ is one, and that for each $\pi'_s(s')$, it is true that $0 \leq \pi'_s(s') \leq 1$.

$$\sum_{s' \in G} \pi'_s(s') = D^{c'+2} \left(\sum_{s' \in G} \pi_s(s') - \left(1 - \frac{1}{D^{c'+2}}\right) \sum_{s' \in G} \pi(s')\right) = D^{c'+2} - D^{c'+2} + 1 = 1.$$

It remains to show that for each $s' \in G$, it is true that $0 \leq \pi'_s(s') \leq 1$. For $s' \in G$:

$$\begin{aligned} \pi'_s(s') &= D^{c'+2} \left(\pi_s(s') - \left(1 - \frac{1}{D^{c'+2}}\right) \pi(s')\right) \\ &\leq D^{c'+2} \|\pi_s - \pi\| + \pi(s') \\ &\leq \frac{1}{D^{d-c'-2}} + 1 - \frac{1}{D^{c''}} \leq 1. \end{aligned}$$

Here we use the fact that, by Corollary 2.4.6, π_s and π are D-approximately equivalent, the bound on $\pi(s')$ from Lemma 2.4.5, and the assumptions $d > 2(c'+1)$

and $c'' = 2^{-f_1(D)} < 1$. Similarly,

$$\pi'_s(s') \geq \frac{\pi(s')}{D^{c'+2}} - D^{c'+2}\|\pi_s - \pi\| \geq \frac{1}{D^{c''}D^{c'+2}} - \frac{1}{D^{d-c'-2}} \geq 0,$$

where the last step follows since $d > 2(c'+2)$ and $c'' = 2^{-f_1(D)} < 1$. □

We now show that within each class B_i, approximating the random walk of an agent in the Markov chain by drawing its state after $r \in B_i$ rounds *independently* from the stationary distribution π does not introduce a substantial error. Consider the following modification to the original Markov chain.

Consider a modified Markov chain M in which we add two auxiliary states A_s and B_s for each state s of the original Markov chain, such that the transition from s to A_s is with probability $1 - 1/D^{c'}$ and the transition from s to B_s is with probability $1/D^{c'}$. Additionally, all other outgoing transitions from state s in the original Markov chain are removed. From state A_s we add transitions to other states according to π, and from B_s we add transitions to other states according to π'_s (see Figure 2-4).

In the original Markov chain, for each round $r \in B_i$, immediately after which the agent is in state s, the state immediately after round $r+\beta$ is determined based on the distribution π_s. In the modified Markov chain M, the state immediately after round $r+\beta$ is determined by π from state A_s, and by π'_s from state B_s. By Lemma 2.4.7, it is clear that the distribution of states visited in rounds $r \in B_i$ in the original Markov chain is the same as the distribution in the corresponding rounds of the modified Markov chain.

Let $\mathcal{E}_i$ denote the event that for all rounds $r \in B_i$ and $r \leq \Delta$ in which the Markov chain M is in some state s, the next state reached from s is state A_s (so the state immediately after round $r + \beta$ is chosen from π).

Corollary 2.4.8. *For each i, $1 \leq i \leq \beta$, $P[\mathcal{E}_i] \geq 1 - 1/D^{c'}$.*

Proof. Consider the coin flips in all rounds $r \in B_i$ in which the Markov chain M is in some state s; these coin flips determine whether the next state is A_s or B_s. By the definition of the modified Markov chain M, with probability at least $1 - 1/D^{c'}$, the

next state is A_s. By a union bound, the probability that the next state is A_s for all rounds $r \in B_i$ (whose number is Δ/β where $\beta = D^{o(1)} < D^2$) is:

$$1 - \frac{\Delta}{D^{c'+2}} \geq 1 - \frac{1}{D^{c'+2-2+f_2(D)}} \geq 1 - \frac{1}{D^{c'}},$$

where the last step follows since $f_2(D)$ is positive. Therefore, $P[\mathcal{E}_i] \geq 1 - 1/D^{c'}$. $\square$

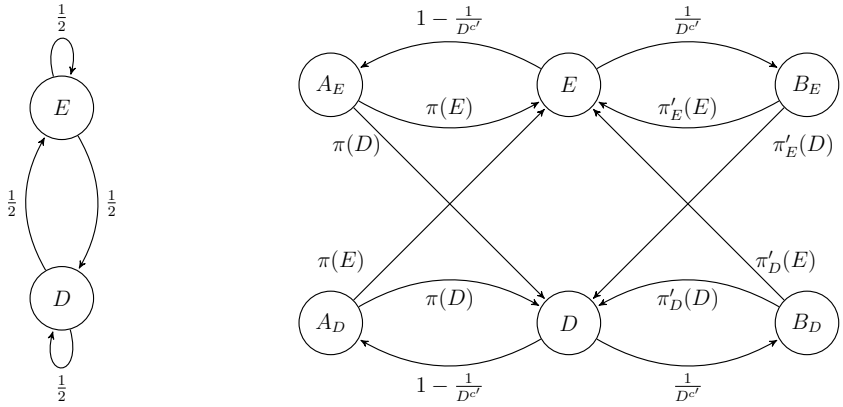

Figure 2-4: On the left: equivalence class $\{E, D\}$ induced by P^2 from Figure 2-3. On the right: derived Markov chain M, ignoring the exact probabilities on the left.

In this section, we showed that the distribution that determines an agent's behavior is very close to the stationary distribution of the recurrent class in which the agent is located. In the next section, we will use this result to argue that if the agent does not explore the grid well when behaving according to the stationary distribution, then it does not explore the grid considerably better when behaving according to the actual distribution of the algorithm.

Movement on the grid

Next, we focus on the implications of the results in the previous sections on the agents' movement in the grid. In order to use Corollary 2.4.8, we will base the results of this subsection on the behavior of the derived Markov chain M. However, since we are only reasoning about rounds from blocks B_i for some i, as we already mentioned, by Lemma 2.4.7, the distribution of states in the derived Markov chain M is the same as

in the original Markov chain. Therefore, the results about the movement of the agents on the grid based on Markov chain M also apply to the movement of the agents on the grid in the original Markov chain.

Let indicator random variable $X_r^{\uparrow}$ have value 1 if the state of the agent after r rounds is labeled *up*, and 0 otherwise. Note that these random variables depend only on the state transitions the agent performs in the derived Markov chain M. Also let $X_{\leq r}^{\uparrow} = \sum_{r'=1}^{r} X_{r'}^{\uparrow}$ denote the total number of steps *up* in the grid up to round r. Similarly, we can define random variables $X_{\leq r}^{\rightarrow}$, $X_{\leq r}^{\downarrow}$, and $X_{\leq r}^{\leftarrow}$ to refer to the number of steps $right$, *down*, and $left$ in the grid up to round r.

Recall that $\mathcal{E}_i$ denotes the event that for all rounds $r \in B_i$, the state in Markov chain M immediately after round $r+\beta$ is drawn from the stationary distribution. By Corollary 2.4.8, $\mathcal{E}_i$ occurs with probability at least $1 - 1/D^{c'}$.

First, we show that, with high probability, for all rounds $r \in B_i$, the number of moves *up* of the agent in those rounds does not differ by more than $o(D/(|S|\beta))$ from the expected number of such moves conditioning on event $\mathcal{E}_i$. Denote by $p_i^{\uparrow}$ the probability for the agent to move up when its state is distributed according to π.

Lemma 2.4.9. *For each i, where $1 \leq i \leq \beta$, and each round $r \leq \Delta$, conditioning on event $\mathcal{E}_i$, with probability at least $1 - 1/D^{c'-1}$, it is true that:*

$$\left| \sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} X_{r'}^{\uparrow} - \mathbb{E}\left[\sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} X_{r'}^{\uparrow} \,\middle|\, \mathcal{E}_i \right] \right| = o\left(\frac{D}{|S|\beta} \right).$$

Proof. Conditioned on $\mathcal{E}_i$, we know that the considered variables $X_{r'}^{\uparrow}$ from B_i are independently and identically distributed: The state after r' rounds is drawn independently from some stationary distribution π that does not depend on r', and the probability for the agent to move up in the grid equals the probability that this state is labeled *up*.

By linearity of expectation,

$$\mu_i = \mathbb{E}\left[\sum_{\substack{\beta+1\le r'\le r\\ r'\in B_i}} X^{\uparrow}_{r'} \,\middle|\, \mathcal{E}_i\right] = \sum_{\substack{\beta+1\le r'\le r\\ r'\in B_i}} \mathbb{E}\left[X^{\uparrow}_{r'} \,\middle|\, \mathcal{E}_i\right] = \sum_{\substack{\beta+1\le r'\le r\\ r'\in B_i}} p^{\uparrow}_i = p^{\uparrow}_i \left\lfloor \frac{r}{\beta} - 1 \right\rfloor.$$

Next, we would like to apply a Chernoff bound to the random variable with expectation μ_i. Technically, we need to consider two cases, depending on whether $\mu_i \le 3c' \ln D$ or not. Instead, for simplicity, we will define a new random variable $Z^{\uparrow}_r$ that captures both of these cases.

Let $Y^{\uparrow}_y$ be a binary random variable such that for each $1 \le y \le \lceil 3c' \ln D - \mu_i \rceil$:

$$P[Y^{\uparrow}_y = 1] = \frac{\max\{0, 3c' \ln D - \mu_i\}}{\lceil 3c' \ln D - \mu_i \rceil}.$$

Also, note that for all $1 \le y \le \lceil 3c' \ln D - \mu_i \rceil$ the $Y^{\uparrow}_y$ variables are identical and independent.

Let $Z^{\uparrow}_r$ be a random variable such that:

$$Z^{\uparrow}_r = \sum_{\substack{\beta+1\le r'\le r\\ r'\in B_i}} X^{\uparrow}_{r'} \,\Big|\, \mathcal{E}_i + \sum_{y=1}^{\lceil 3c' \ln D - \mu_i \rceil} Y^{\uparrow}_y.$$

By linearity of expectation:

$$E[Z^{\uparrow}_r] = \mu_i + \lceil 3c' \ln D - \mu_i \rceil \cdot \frac{\max\{0, 3c' \ln D - \mu_i\}}{\lceil 3c' \ln D - \mu_i \rceil} = \max\{\mu_i, 3c' \ln D\}.$$

Now, we can see that by defining the random variables $Y^{\uparrow}_y$ in the specific way we did, the random variable $Z^{\uparrow}_r$ has the expectation that we need: the maximum of the expectation we care about (μ_i) and the threshold value $3c' \ln D$.

By a Chernoff bound (Theorem A.1.5) with $\delta = \sqrt{3c' \ln D / E[Z^{\uparrow}_r]}$, it follows that:

$$P\left[\left|Z^{\uparrow}_r - E[Z^{\uparrow}_r]\right| > \delta E[Z^{\uparrow}_r]\right] \le 2e^{-\delta^2 \mu_i/3} = \frac{2}{D^{c'}} \le \frac{1}{D^{c'-1}}.$$

If $E[Z_r^{\uparrow}] = \mu_i$, using the fact that $r \leq \Delta = o(D^2/(\beta|S|^2 \log D))$, we get:

$$\delta E[Z_r^{\uparrow}] = \sqrt{3c' \ln D\mu_i} = \mathcal{O}\left(\sqrt{\frac{p_i^{\uparrow} \Delta \log D}{\beta}}\right) = o\left(\frac{D}{|S|\beta}\right).$$

Otherwise, if $E[Z_r^{\uparrow}] = 3c' \ln D$, then $\delta E[Z_r^{\uparrow}] = 3c' \ln D = o(D/(|S|\beta))$. Since we are considering the number of moves *up* in the grid, we know that:

$$P\left[\sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} X_{r'}^{\uparrow} \geq 0 \,\middle|\, \mathcal{E}_i\right] = 1.$$

Therefore, in either case, we conclude that, with probability at least $1 - 1/D^{c'-1}$:

$$\left|\sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} X_{r'}^{\uparrow} - \mathbb{E}\left[\sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} X_{r'}^{\uparrow} \,\middle|\, \mathcal{E}_i\right]\right| = o\left(\frac{D}{|S|\beta}\right).$$

□

Next, we show that, with high probability, for *all* rounds up to round r (not just the rounds $r \in B_i$), the number of moves *up* performed by the agent does not differ by more than $o(D/|S|)$ from some fraction of r.

Lemma 2.4.10. *There exists $p^{\uparrow} \in [0,1]$, such that for each round $r \leq \Delta$, with probability at least $1 - 1/D^{c'-2}$, it holds that $\left|X_{\leq r}^{\uparrow} - rp^{\uparrow}\right| = o(D/|S|)$.*

Proof. Recall that for each i, where $1 \leq i \leq \beta$, B_i is the collection of step numbers $i + j\beta \leq \Delta$ for $j \in \mathbb{N}_0$. Therefore:

$$\sum_{r'=\beta+1}^{r} X_{r'}^{\uparrow} = \sum_{i=1}^{\beta} \sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} X_{r'}^{\uparrow}.$$

By Lemma 2.4.9, we know that for each i, conditioned on $\mathcal{E}_i$, with probability at

least $1 - 1/D^{c'-1}$ it holds that:

$$\left| \sum_{\substack{\beta+1 \le r' \le r \\ r' \in B_i}} X^{\uparrow}_{r'} - \mathbb{E}\left[\sum_{\substack{\beta+1 \le r' \le r \\ r' \in B_i}} X^{\uparrow}_{r'} \,\middle|\, \mathcal{E}_i \right] \right| = o\left(\frac{D}{|S|\beta}\right)$$

To complete our line of reasoning, we need to incorporate the preceding β rounds as well. Note that the expected number of *up* moves in β rounds is at most β. Since the modified Markov chains corresponding to each block of rounds B_i are independent from each other, it follows that:

$$\mathbb{E}\left[X^{\uparrow}_{\le r} \,\middle|\, \bigwedge_{i=1}^{\beta} \mathcal{E}_i \right] \le \sum_{i=1}^{\beta} \mathbb{E}\left[\sum_{\substack{\beta+1 \le r' \le r \\ r' \in B_i}} X^{\uparrow}_{r'} \,\middle|\, \mathcal{E}_i \right] + \beta = r \sum_{i=1}^{\beta} \frac{p^{\uparrow}_i}{\beta} + \beta$$

Setting $p^{\uparrow} = \sum_{i=1}^{\beta} p^{\uparrow}_i / \beta$, the above expectation is at most $rp^{\uparrow} + \beta$.

By a union bound, $\bigwedge_i \mathcal{E}_i$ occurs with probability at least $1 - 1/D^{c'-1}$ because there are $\beta = o(D)$ such events and each one of them holds with probability at least $1 - 1/D^{c'}$, by Corollary 2.4.8. By another union bound, with probability at least $1 - 1/D^{c'-2}$, both $\bigwedge_i \mathcal{E}_i$ occurs and Lemma 2.4.9 holds for all i. By the definition of β, it follows that $\beta = o(D/|S|)$. Also, since, the expected number of moves *up* in β rounds is at most β, and the actual number of such moves differs by at most β from the expectation, it follows that:

$$\begin{aligned} \left| X^{\uparrow}_{\le r} - rp^{\uparrow} \right| &= \left| X^{\uparrow}_{\le r} - \mathbb{E}\left[X^{\uparrow}_{\le r} \,\middle|\, \bigwedge_{i=1}^{\beta} \mathcal{E}_i \right] \right| + \left| \mathbb{E}\left[X^{\uparrow}_{\le r} \,\middle|\, \bigwedge_{i=1}^{\beta} \mathcal{E}_i \right] - rp^{\uparrow} \right| \\ &\le \sum_{i=1}^{\beta} o\left(\frac{D}{|S|\beta}\right) + \beta + \beta = o\left(\frac{D}{|S|}\right). \end{aligned}$$

□

We can repeat these arguments for the other directions (right, down, and left).

Corollary 2.4.11. *1. There exists $p^{\rightarrow} \in [0,1]$, such that for each round $r \le \Delta$, with probability at least $1 - 1/D^{c'-2}$, it holds that $\left| X^{\rightarrow}_{\le r} - rp^{\rightarrow} \right| = o(D/|S|)$.*

2. *There exists* $p^{\downarrow} \in [0,1]$*, such that for each round* $r \leq \Delta$*, with probability at least* $1 - 1/D^{c'-2}$*, it holds that* $\left|X^{\downarrow}_{\leq r} - rp^{\downarrow}\right| = o(D/|S|)$.

3. *There exists* $p^{\leftarrow} \in [0,1]$*, such that for each round* $r \leq \Delta$*, with probability at least* $1 - 1/D^{c'-2}$*, it holds that* $\left|X^{\leftarrow}_{\leq r} - rp^{\leftarrow}\right| = o(D/|S|)$.

Define $X_{\leq r} \in \mathbb{Z}^2$ to be the random variable describing the sum of all moves the agent performs in the grid up to round r, i.e., its position in the grid (in each dimension) after r rounds. For this random variable, we show that the position of the agent after r rounds does not differ by more than $o(D/|S|)$ from some fraction of r.

Corollary 2.4.12. *There exists* $\vec{p} \in [-1,1]^2$*, such that for each* $r \leq \Delta$*, with probability at least* $1 - 1/D^{c'-3}$*,* $\|X_{\leq r} - r\vec{p}\| = o(D/|S|)$.

Proof. Observe that $X_{\leq r} = (X^{\uparrow}_{\leq r} - X^{\downarrow}_{\leq r}, X^{\rightarrow}_{\leq r} - X^{\leftarrow}_{\leq r})$. Hence, setting $\vec{p} = (p^{\uparrow} - p^{\downarrow}, p^{\rightarrow} - p^{\leftarrow})$, by Lemma 2.4.10, Corollary 2.4.11 and a union bound, it follows that $\|X_{\leq r} - r\vec{p}\| = o(D/|S|)$ with probability at least $1 - 1/D^{c'-3}$. □

We are now ready to resume the proof of Theorem 2.4.1.

Proof of Theorem 2.4.1. Let $\mathcal{C}$ be the set of recurrent classes of the original Markov chain of each agent. By Corollary 2.4.3, it holds for each agent that, with probability at least $1 - 1/D^{c'-3}$, the agent is located in some recurrent class $C(a) \in \mathcal{C}$ within R_0 rounds. By Corollary 2.4.4, with probability at least $1 - 1/D^{c'-3}$, either the agent visits a state labeled origin every R_0 rounds, or none of the states in C are labeled *origin*. In the first case, then the agent does not visit a point in the grid at distance more than $R_0 = D^{o(1)}$ from the origin. In the second case, we can apply Corollary 2.4.12 to conclude that, with probability at least $1 - 1/D^{c'-3}$, the position of the agent does not deviate by more than distance $o(D/|S|)$ from a straight line in the grid starting at the origin and ending at point $\Delta\vec{p}$ ($\vec{p}$ depends only on $C(a)$). Therefore, by a union bound with the results from Corollary 2.4.3, it follows that with probability at least $1 - 1/D^{c'-4}$, either an agent does not venture further away from the origin than distance $o(D/|S|)$, or its position does not deviate by more than distance $o(D/|S|)$ from one of at most $|\mathcal{C}|$ straight lines or the origin. By a union

bound, this holds for all agents jointly with probability at least $1-1/D^{c'-4-c_n} = 1/D^c$ (recall that by assumption $n \leq T(D) \leq D^{c_n}$).

Since for any straight line only a segment of length $\mathcal{O}(D)$ is in distance $\mathcal{O}(D)$ from the origin, the union of all grid points that are (i) in distance at most D from the origin and (ii) in distance at most $o(D/|S|)$ from one of the $|\mathcal{C}|$ straight lines has cardinality $\mathcal{O}(D) \cdot o(D/|S|) \cdot |\mathcal{C}| \leq o(D^2/|S|) \cdot |S| = o(D^2)$. Hence, there is a set $G \subset \mathbb{Z}^2$ of $o(D^2)$ grid points that only depends on the algorithm $\mathcal{A}_D$ such that, with probability at least $1 - 1/D^c$, all grid points in distance D from the origin that are visited within the first $R_0 + \Delta$ steps of an execution of $\mathcal{A}_D$ are in G. Since there are $\Theta(D^2)$ grid points in distance D from the origin, this implies that the target can be placed in such a way that, with probability at least $1-1/D^c$, no agent will find it. $\square$

In the above proof, note that if the target is placed uniformly at random in the square with side length $2D$ centered at the origin, then it is no longer true that no algorithm finds it in the specified amount of time. In fact, any algorithm that explores at least one grid point has a $\Omega(1/D^2)$ probability of finding a uniformly-placed target. Thus, the correctness of Theorem 2.4.1 relies on the fact that the target is placed in the grid adversarially.

2.4.3 Theorem for M_{moves} and non-uniform algorithms

First, we show that Theorem 2.4.1 also holds with respect to the metric M_{moves}. In the following corollary, we show that either each move of an agent on the grid corresponds to at most $D^{o(1)}$ transitions in its Markov chain, or the agent does not move on the grid after some point on. Therefore, since Theorem 2.4.1 guarantees that, with high probability, no agent finds the target in $D^{2-o(1)}$ *steps*, then it must be true that, with high probability, no agent finds the target in $D^{2-o(1)}$ *moves*.

Fix a constant $c > 0$ and let $f_3 : \mathbb{Z}^+ \to [1, \infty)$ be an arbitrary function such that $f_3(D) = o(1)$ and $f_3(D) \leq f_2 - 2^{-f_1(D)} + 3 \log\log D / \log D$ for any D. Recall that T is an arbitrary polynomial such that $T(D) \leq D^{c_n}$ for any D.

Corollary 2.4.13. *For each $D \in \mathbb{N}$, $D > 1$ and each $n \in \mathbb{N}$, $n \leq T(D)$, assume algorithm $\mathcal{A}_D$ with n agents satisfies $\chi(\mathcal{A}_D) = b + \log \ell \leq \log \log D - f_1(D)$. Then, there exists a placement (x, y), $|x|, |y| \leq D$ of the target, such that, with probability at least $1 - 1/D^c$, algorithm $\mathcal{A}_D$ satisfies $M_{moves} > D^{2-f_3(D)}$ for this placement (x, y).*

Proof. The setup for this proof is the same as in Theorem 2.4.1, so we use the same constants and other values defined in Section 2.4.2. Also, the results from Section 2.4.2 hold with respect to these constants and values, so we can reuse them here.

Consider any fixed execution prefix of length R_0 rounds in which an agent is in some state s in some recurrent class C. By Corollary 2.4.3, this is true with probability at least $1 - 1/D^{c'}$. If C contains only states labeled *none*, then the agent does not make any progress in the grid after it reaches its recurrent class, so it does not visit more than R_0 grid points.

Otherwise, if C contains a state s', labeled *up*, *down*, $left$, or $right$, we show that it is reachable from state s after r rounds such that $R_0 \leq r \leq 2R_0$. By Lemma 2.4.2, with probability at least $1 - 1/D^{c'}$, the agent visits state s' within R_0 rounds. In an execution of length $R_0 + \Delta$, there are $o(D^2)$ groups of R_0 rounds. By a union bound, with probability at least $1 - 1/D^{c'-2}$, the agent visits a state labeled *up*, *down*, $left$, or $right$ at least Δ/R_0 times. By the law of total probability, since all execution prefixes of length R_0 are disjoint, this conclusion holds for all executions. By a union bound, this result and Corollary 2.4.3 hold jointly with probability at least $1 - 1/D^{c'-3}$.

By Theorem 2.4.1, there is a placement of the target such that, with probability at least $1 - 1/D^{c'-4}$, no agent finds it within $D^{2-f_2(D)}$ steps. With probability at least $1 - 1/D^{c'-3}$, R_0 steps correspond to at least one move. By a union bound, with probability at least $1 - 1/D^{c'-4}$, $\Delta = D^{2-f_2(D)}$ steps correspond to at least $\Delta/R_0 = D^{2-f_2(D)}/R_0 \geq D^{2-f_3(D)}$ moves. Therefore, no agent finds the target in $D^{2-f_3(D)}$ moves with probability at least $1 - 1/D^{c'-5} \geq 1 - 1/D^c$. □

2.4.4 Theorem for M_{moves} and uniform algorithms

Finally, we extend Corollary 2.4.13 to also hold for uniform algorithms.

Corollary 2.4.14. *For any $D \in \mathbb{N}$, $D > 1$, any $n \in \mathbb{N}$, $n \leq T(D)$, and any uniform algorithm $\mathcal{A}$ with n agents, assume that $\chi(\mathcal{A}) = b + \log \ell \leq \log \log D - f_1(D)$. Then, there exists a placement (x, y), $|x|, |y| \leq D$ of the target, such that, for any constant $c > 1$, with probability at least $1 - 1/D^c$, algorithm $\mathcal{A}$ satisfies $M_{moves} > D^{2-f_3(D)}$ for this placement (x, y).*

Proof. Note that the proofs of Theorem 2.4.1 and Corollary 2.4.13 are with respect to the Markov chain induced by the non-uniform algorithm. This Markov chain may have information about D encoded in it but throughout the proofs, the only way the value of D is used is through the constraint on the selection metric $\chi = b + \log \ell \leq \log \log D - f_1(D)$. Therefore, even if we consider a uniform algorithm, instead of a non-uniform algorithm, the results and the proofs still hold because the same restriction on χ applies to the uniform algorithm. □

Finally, note that following similar reasoning, we can show that the lower bound holds for algorithms uniform and non-uniform in n because the only restriction we use is on χ which is not related to n.

2.5 Discussion

One contribution of our work on foraging is considering a new compound metric, χ, which captures the nature of the search problem more comprehensively compared to the standard metrics of time and space complexity. The χ metric does include components of the space metric (the number of bits each agent is allowed to use), however, it combines this standard metric with a measure of the range of probability values an algorithm is allowed to use. As mentioned in Chapter 1, these two metrics are related to each other in that more bits allow us to implement coins with more bias, and also sufficiently biased coins can give algorithms similar power as large memory. Instead of considering the memory and probability metrics separately, and potentially analyzing a fixed trade-off between them, we combine them in a single metric in order to cover the entire range of trade-offs between the memory and probability

range components. The goal is for our results to hold regardless of how an algorithm chooses to trade off the two components of the χ metric.

The main contribution of our results is establishing $\chi \approx \log\log D$ as the threshold for an algorithm to search the plane efficiently. In particular, we consider efficiency in terms of the speed-up as the number of searchers n increases. As mentioned in Chapter 1, simple random walks are not very efficient in terms of exploring the plane in parallel; in particular, n random walks speed up the search process of a single random walk only by a factor of $\log n$, where the optimal and desired speed-up is linear in n. For comparison, our lower bound indicates that any algorithm with a χ value less than the $\log\log D - \omega(1)$ threshold cannot search the plane significantly faster than n random walks. On the other hand, our algorithms indicate that a χ value of $\mathcal{O}(\log\log D)$ is sufficient to get the optimal speed-up of $\Omega(n)$ and also achieve the optimal running time of $\Theta(D^2/n + D)$ for searching the plane.

2.6 Open Problems

Various aspects of the algorithms presented in the previous sections can be improved. For example, as mentioned earlier, we can make the algorithms also uniform in n by following the strategy in [51]; this modification will result in a $O(\log n)$ factor overhead in the running time. Furthermore, since our algorithms do not rely on communication or carefully synchronized rounds, it seems natural to analyze the fault tolerance properties the algorithms satisfy. We believe the correctness of the algorithms will not be affected by faults, as long as the faults are not adversarially targeted at a specific area of the grid; for example, if each agent that gets within distance of one hop to the target is crashed, then clearly our algorithms (or any other algorithms) cannot guarantee anything. Finally, it would also be interesting to consider various forms of communication between the agents and analyze the properties of the resulting algorithms. A recent attempt at understanding such behavior is [84], where the agents use only a loneliness detection capability in order to explore the grid with constant memory and constant probabilities.

Another potential extension of our work includes using the techniques from our lower bound to prove lower bounds with similar restrictions in other graphs. For example, consider multiple non-communicating agents with limited memory and probabilities trying to explore a tree or some other data structure with some regularity properties. Such a result may be useful in designing systems where a data structure needs to be explored by multiple threads without the need (or capability) to support inter-thread communication.

Foraging in general is a very rich problem that can be studied in various settings and with a range of different assumptions. For example, besides from the exploration phase in which ants search for a target, it is interesting to also consider the exploitation phase of foraging in which ants bring items back to the nest. One direction is to explore different pheromone structures (like trees rooted at the nest leading to various food sources) that ants use in order to retrieve food. Pheromones represent a communication capability that is also interesting to study from the point of view of the power it gives algorithms and also the cost associated with using it (the production and dispersion of pheromones).

Finally, a natural extension to the general foraging problem is to consider multiple targets and potentially targets that appear and disappear dynamically throughout the execution. Some assumptions include choosing a distribution according to which the targets appear and disappear, spatially and temporally, or considering an adversary who determines when and where this happens. Also, this extension of the problem may require different metrics to analyze the efficiency of algorithms, for example, the rate at which food is brought back to the nest in terms of the rate at which it appears and disappears.

Chapter 3

House Hunting

In this chapter, we study the house hunting problem, in which an ant colony needs to reach consensus on a new nest for the colony to move to. The main results of the chapter are a mathematical model of the house-hunting process, a lower bound on the number of rounds required by any algorithm solving the house-hunting problem in the given model, and two house-hunting algorithms.

The model (Section 3.1) is based on a synchronous model of execution with n probabilistic ants and communication limited to one ant leading a randomly chosen ant (tandem run or transport) to a candidate nest. Ants can also search for new nests by choosing randomly among all k candidate nests.

The lower bound (Section 3.2) states that, under this model, no algorithm can solve the house-hunting problem in time sub-logarithmic in the number of ants. The main proof idea is that, in any step of an algorithm's execution, with constant probability, an ant that does not know of the location of the eventually-chosen nest remains uninformed. Therefore, with high probability, $\Omega(\log n)$ rounds are required to inform all n ants. This technique closely resembles lower bounds for rumor spreading in a complete graph, where the rumor is the location of the chosen nest [73].

The first algorithm (Section 3.3) solves the house-hunting problem in asymptotically optimal time. The main idea is a typical example of positive feedback: each ant leads tandem runs to some suitable nest as long as the population of ants at that nest keeps increasing; once the ants at a candidate nest notice a decrease in the popula-

tion, they give up and wait to be recruited to another nest. With high probability, within $\mathcal{O}(\log n)$ rounds, this process converges to all n ants committing to a single winning nest. Unfortunately, this algorithm relies heavily on a synchronous execution and on the ability to precisely count nest populations, suggesting that the algorithm is susceptible to perturbations of our model and most likely does not match real ant behavior.

The goal of the second algorithm (Section 3.4) is to be more natural and resilient to perturbations of the environmental parameters and ant capabilities. The algorithm uses a simple positive-feedback mechanism: in each round, an ant that has located a candidate nest recruits other ants to the nest with probability proportional to its current population. We show that, with high probability, this process converges to all n ants being committed to one of the k candidate nests within $\mathcal{O}(k^3 \log^{1.5} n)$ rounds. While the algorithm's complexity analysis does not match the lower bound, the algorithm exhibits a much more natural process of converging to a single nest. In Section 3.5, we discuss in more detail how to combine this algorithm with a subroutine that provides ants with an estimate of the population of a candidate nest. We show that the algorithm remains correct under such uncertainty on the environment. Such robustness criteria are necessary in nature and generally desirable for distributed algorithms.

3.1 Model

Here we present a simple model of *Temnothorax* ants behavior that is tractable to rigorous analysis, yet rich enough to provide a starting point for understanding real ant behavior.

The environment consists of a home nest, denoted n_0, along with k candidate new nests, identified as n_i for $i \in \{1, \cdots, k\}$. Each nest n_i is assigned a quality $q(i) \in Q$, from some set Q. Throughout this chapter we let $Q = \{0, 1\}$, with quality 0 indicating an unsuitable nest, and 1 a suitable one. Additionally, we assume that there is always at least one nest with $q(i) = 1$.

The colony consists of n identical probabilistic finite state machines, representing the ants. We assume n is somewhat larger than k ($k = \mathcal{O}(n/\log n)$). Also, we assume that the ants do not know the value of k but they know the value of n. This last assumption is based on evidence that real *Temnothorax* ants and other species are able to estimate the size of the colony [30, 39].

The general behavior of the state machines is unrestricted but their interactions with the environment and with other ants are limited to the high-level functions **search**(), **go**(), and **recruit**(), defined below.

We assume a synchronous model of execution, starting at time 0 when all the ants are located at the home nest. Each round $r \geq 1$ denotes the transition from time $r-1$ to time r. At each time r, each ant a is located at a nest, denoted by $\ell(a, r) \in \{0, 1, \cdots, k\}$. We assume that for each ant a, $\ell(a, 0) = 0$. For $r \geq 1$, the value of $\ell(a, r)$ is set by the calls to **search**(), **go**(), or **recruit**() made by the ant in round r according to the rules below. Also, we assume that an ant a with $\ell(a, r) = i$ and $i \geq 1$ (that is, an ant located at candidate nest n_i) has access to the value of $q(i)$.

Let $c(i, r) = |\{a \mid \ell(a, r) = i\}|$ denote the number of ants located in nest n_i at time r.

In each round r, each ant a performs a call to exactly one of the following functions:

- **search**(): Returns a pair (j, c_j), where $j \in \{1, \cdots, k\}$ is a nest index, and c_j is a positive integer. Sets $\ell(a, r) := j$. Index j is chosen uniformly at random from $\{1, \cdots, k\}$. This function represents ant a searching for a nest and returns the nest index of a randomly chosen nest and the number of ants at that nest.

- **go**(**i**): Takes as input the index $i \in \{0, \cdots, k\}$ of a nest, and returns a pair (j, c_j), where $j \in \{0, \cdots, k\}$ is a nest index and c_j is a positive integer. Sets $\ell(a, r) := j$. Index i is such that there exists a time $r' \leq r$ in which $\ell(a, r') = i$. Also, we require that $j = i$. The function represents ant a revisiting a candidate nest n_i and returns the number of ants at nest n_i at time r (as well as index i).

- **recruit**(**b**, **i**): Takes as input a boolean $b \in \{0, 1\}$ and a nest index $i \in \{1, \cdots, k\}$, and returns a pair (j, c_j), where $j \in \{1, \cdots, k\}$ is a nest index,

and c_j is a positive integer[1]. Sets $\ell(a, r) := j$. The nest index i is such that there exists a time $r' \leq r$ in which $\ell(a, r') = i$.

The recruitment strategy is defined in Algorithm 5.

In the recruitment strategy, the goal is for each actively recruiting ant to choose a random ant to recruit. Ants do so by first forming a random permutation P. Then, in the order of the permutation, each recruiting ant (with input $b = 1$) chooses a uniformly random ant to recruit. The recruitment is successful if the chosen ant has not already recruited an ant itself and has not been recruited by another ant. In Algorithm 5, we use the set R of all ants that call **recruit**$(\cdot, \cdot)$, the set $S \subseteq R$ of ants that call **recruit**$(\mathbf{1}, \cdot)$ and the random permutation P of all ants in R. Then, using the simple sequential rule described above, we add ants to the set M of all successful recruitment pairs, and to the set N of all ants that participate in successful recruitment pairs. Finally, we assign the return value j to each ant as follows: the first ant a' in each pair in M and any ant not present in N receive the same output j as their input i; the second ant a in any pair $(a', a) \in M$ receives as output j the nest index i from the input to **recruit**$(\cdot, \mathbf{i})$ by ant a'.

The permutation P simply serves as tie-breaker to avoid conflicts between recruitments. It is important to note that this process is not a distributed algorithm executed by the ants, but just a modeling tool to formalize the idea of ants recruiting other ants randomly without introducing any inconsistencies between the ordered pairs of recruiting and recruited ants. Algorithm 5 can be thought of as a centralized process run by the environment that places ants temporarily (for the duration of the round) in the home nest and pairs them appropriately. We believe our results also hold under other natural models for randomly pairing ants. For example, another way to pair ants is to let each recruiting ant choose a uniformly random ant and consider the recruitment successful only if the chosen ant is not recruiting as well. This rule results in fewer recruitment pairs compared to Algorithm 5 because it does not allow for a recruiting ant to choose another recruiting ant. The algorithms in this

[1]Note that an ant is not allowed to recruit to the home nest (corresponding to input $i = 0$).

chapter can easily be adapted to work with this alternative recruitment rule without affecting their asymptotic running time.

Algorithm 5: Generate return values j for all ants that call **recruit**$(\cdot, \cdot)$.

```
R: the set of ants that call recruit(·,·)
P: a uniform random permutation of all ants in R (P : ℕ → R)
P(x): x'th ant in P, for x ∈ {1, ··· , |P|}
S: the set of ants S ⊆ R that call recruit(1,·)
M: a set of ordered pairs of ants, initially ∅
N: a set of ants, initially ∅
for x = 1 to |P| do
    if P(x) ∈ S \ N then
        y := uniform random integer in {1, ··· , |P|}
        if P(y) ∉ N then
            M := M ∪ (P(x), P(y))
            N := N ∪ {P(x), P(y)}
for x = 1 to |P| do
    if ∃y, (P(y), P(x)) ∈ M then
        return j to ant P(x) where j is input to recruit(·, j) called by P(y)
    else return j to ant P(x) where j is input to recruit(·, j) called by P(x)
```

Finally, we need to specify the c_j components of the return pairs for the functions above. Note that each function returns a pair (j, c_j) with various restrictions on the nest index j, determined by each function separately. Once the $\ell(a, r)$ values have been set for each ant a in round r, for each return pair (j, c_j), we let $c_j = c(j, r)$.

An ant *recruits successfully* if it is the recruiting ant (first element) in one of the pairs in M.

Our model for recruitment encompasses both the tandem runs and direct transport behavior observed in *Temnothorax* ants. Since direct transport is only about three times faster than tandem walking [93], and since we focus on asymptotic behavior, we do not model these two types of actions separately.

Next, we prove a general statement about the recruitment process that will be used in the proofs of our lower bound and algorithms.

Lemma 3.1.1. *Let a be an arbitrary ant that executes recruit$(1, \cdot)$ in some round r. Then, with probability at least $1/16$, for some ant a', $(a, a') \in M$.*

Proof. Let R denote the set of ants that call $recruit(\cdot,\cdot)$ in round r, and let P be the random permutation of ants in R. The probability distribution we consider is over the permutation P and the random choices of the ants in round r. Fix an arbitrary ant a that calls **recruit**$(\mathbf{1},\cdot)$ in round r, and an arbitrary constant $c > 1$.

Let E denote the event that for some ant a', $(a, a') \in M$; that is, ant a successfully recruits some ant in round r.

First, note that if $|R| < 2$, then ant a is forced to recruit itself, so $\Pr[E] = 1$. For the rest of the proof we assume $|R| \geq 2$.

Let E' denote the event that ant a is located in the first half of P and a' is located in the second half of P. More precisely, let the first half refer to ants in positions 1 to $\lceil |R|/2 \rceil$, and let the second half refer to ants in positions $\lceil |R|/2 \rceil$ to $|R|$ (note that the two halves overlap by one ant). So $\Pr[E'] \geq 1/4$.

By the definition above, conditioning on event E', there are at most $\lceil |R|/2 \rceil - 1$ ants in P before ant a. Also, by the definition of the recruitment process, the probability that a fixed ant chooses another fixed ant is $1/|R|$. Therefore, we have:

$$\begin{aligned}
\Pr[E] &\geq \Pr[E \mid E'] \cdot \Pr[E'] \\
&\geq \left(\frac{1}{4}\right) \Pr[a \text{ and } a' \text{ not recruited successfully by an ant before } a \text{ in } P \mid E'] \\
&\geq \left(\frac{1}{4}\right) \left(1 - \frac{2}{|R|}\right)^{\lceil |R|/2 \rceil - 1} \\
&\geq \left(\frac{1}{4}\right) \left(1 - \frac{2}{|R|}\right)^{|R|/2} \\
&\geq \frac{1}{16}.
\end{aligned}$$

□

Problem Statement: An algorithm $\mathcal{A}$ solves the HouseHunting problem with k nests in $T \in \mathbb{N}$ rounds with error probability δ, for $0 < \delta \leq 1$, if with probability at least $1 - \delta$, taken over all executions of $\mathcal{A}$, there exists a nest $i \in \{1, \cdots, k\}$ such that $q(i) = 1$ and $\ell(a, r) = i$ for all ants a and for all times $r \geq T$.

3.2 Lower Bound

In this section, we present a lower bound on the number of rounds required for an algorithm to solve the house-hunting problem. The key idea of the proof is similar to the lower bounds on spreading a rumor in a complete graph [73] where neighbors contact each other randomly. Assuming a house-hunting process with a single good nest, its location represents the rumor to be spread among all ants and communication between random neighbors is analogous to the recruiting process.

Assume only a single nest, n_w, has quality 1, and so it is the only option for the ants to relocate to. A lower bound for this particular configuration is sufficient to imply a worst-case lower bound over all nest and quality configurations.

Define an ant a to be *informed* at time r if $\ell(a, r') = w$ for some $r' \leq r$ (ant a has visited the good nest n_w); otherwise, define it to be *ignorant* at time r.

For each ant a, let $\bar{X}_r^a$ be a random variable such that $\bar{X}_r^a = 1$ if ant a is ignorant at time r, and $\bar{X}_r^a = 0$ if ant a is informed at time r.

Lemma 3.2.1. *For $k \geq 2$, for each and a, and each time r,* $\Pr\left[\bar{X}_{r+1}^a = 1 \mid \bar{X}_r^a = 1\right] \geq 1/4$.

Proof. Fix an arbitrary time r and an arbitrary ant a that is ignorant at time r. In round $r+1$, ant a calls exactly one of the three functions: **search**(), **go**($\cdot$), or **recruit**($\cdot, \cdot$). Since a is ignorant at time r, calling **go**($\cdot$) in round $r+1$ implies that a is ignorant at time $r+1$ (because the ant does not visit a new nest).

Suppose a calls **search**(). For $k \geq 2$, the probability that ant a is ignorant at time $r+1$ is $(k-1)/k \geq 1/2$.

Suppose a calls **recruit**($\cdot, \cdot$). Let R be the set of ants that call **recruit**($\cdot, \cdot$) in round $r+1$. Since the number of ants that call **recruit**($\mathbf{1}, \mathbf{w}$) in round $r+1$ (recruiting to the winning nest) is at most $|R|$, and since the probability for a fixed ant to choose another fixed ant in the recruitment process is $1/|R|$, it follows that the probability that ant a is ignorant at time $r+1$ is at least:

$$\left(1 - \frac{1}{|R|}\right)^{|R|} \geq \frac{1}{4} \qquad \text{assuming } |R| \geq 2.$$

Note that if $|R| < 2$, ant a has to recruit itself, so it remains ignorant at time $r + 1$.

Thus, for each possible function that ant a calls in round $r+1$, the probability that it is ignorant at time $r + 1$ is at least $1/4$. Therefore, by the law of total probability, $\Pr\left[\bar{X}^a_{r+1} = 1 \mid \bar{X}^a_r = 1\right] \geq 1/4$. □

Next, we prove a corollary that uses Lemma 3.2.1 to bound the probability that an ant is ignorant at any given time in the execution.

Corollary 3.2.2. *For $k \geq 2$, for each ant a, and each time r, $\Pr\left[\bar{X}^a_r = 1\right] \geq (1/4)^r$.*

Proof. Fix an arbitrary ant a. The proof is by induction on the times in the execution. In the base case, for $r = 0$, the corollary holds because initially all ants are ignorant. Suppose the corollary holds for time r; so, $\Pr\left[\bar{X}^a_r = 1\right] \geq (1/4)^r$. By Lemma 3.2.1, $\Pr\left[\bar{X}^a_{r+1} = 1 \mid \bar{X}^a_r = 1\right] \geq 1/4$. Therefore, $\Pr\left[\bar{X}^a_{r+1} = 1\right] \geq (1/4)^{r+1}$. □

Theorem 3.2.3. *For any constant $c > 1$, let $\mathcal{A}$ be an algorithm that solves the* HouseHunting *problem with $k \geq 2$ nests in T rounds with error probability $1/n^c$. Then, $T = \Omega(\log n)$.*

Proof. Fix an arbitrary constant $c > 1$. Let a be an arbitrary ant and let $r = c \log_4 n$. By Corollary 3.2.2, $\Pr\left[\bar{X}^a_r = 1\right] \geq (1/4)^r = 1/n^c$. Therefore, with probability at least $1/n^c$, ant a is ignorant at time r, so the probability that all ants are informed by time r is at most $1 - 1/n^c$. Since, by assumption, algorithm $\mathcal{A}$ solves the HouseHunting problem in T rounds with error probability $1/n^c$, this implies that $T \geq r = \Omega(\log n)$. □

3.3 Optimal Algorithm

We present an algorithm that solves the HouseHunting problem and is asymptotically optimal. In the first round of the algorithm, each ant searches randomly for a nest. Then, each ant that found a good nest repeatedly tries to recruit other ants to its nest while keeping track of how the population of the nest changes. Nests with a non-decreasing population continue competing while nests with a decreasing population drop out. Due to the uniformity in the recruitment strategy, nests drop out at a constant rate in expectation, meaning that a single winning nest will be identified in $\mathcal{O}(\log k)$ rounds (in expectation and with high probability). Finally, once a single good nest is identified, it takes $\mathcal{O}(\log n)$ rounds (in expectation and with high probability) to recruit all remaining ants to it. Therefore, assuming $k = \mathcal{O}(n)$, the algorithm solves the house hunting problem in asymptotically optimal time (matching the $\Omega(\log n)$ bound in Section 3.2).

This algorithm relies heavily on the synchrony in the execution and the precise counting of the number of ants at a given nest, which makes it sensitive to perturbations of these values, and therefore, not a natural algorithm that resembles real ant behavior. However, the algorithm demonstrates that the HouseHunting problem is solvable in optimal time in the model of Section 3.1.

3.3.1 Algorithm Pseudocode and Description

The pseudocode of the algorithm (Algorithm 6) is slightly more involved than the simple intuitive description above. The additional complexity is necessary for synchronizing the ants in different states (ants from competing nests vs. ants from dropped-out nests) and detecting termination conditions (such as the point in the execution in which a single competing nest remains). First, we present an informal view of the execution of the algorithm, and then we describe the pseudocode in detail through the perspective of a single ant.

Informal Description of the Algorithm Execution

In the first round of the execution of the algorithm, all ants search for nests, and depending on the quality of the nests they find, the ants split into two groups: *active* ants, which found good nests, and *passive* ants, which found bad nests. Since the search process is random, it is possible that all ants arrive at bad nests and enter the passive state (which would cause the algorithm to fail). However, this is very unlikely, as guaranteed by our assumption that $k = \mathcal{O}(n/\log n)$.

After the initial round of searching, active ants may be split between many good nests; we call these *competing* nests. The main goal of the algorithm is to quickly reduce the set of competing nests to a single nest. One straightforward way to achieve this is to somehow let each nest toss an unbiased coin and remain competing if the outcome is heads. Therefore, at each step, about half of the competing nests would drop out, resulting in a single competing nest in $\mathcal{O}(\log k)$ rounds. However, this strategy would require some shared randomness capability of ants at the same competing nest, which may be hard to achieve.

In our algorithm, we try to achieve a similar drop-out rate as above by using a different simple strategy consisting of a sequence of recruitment phases. In each recruitment phase, we let all active ants from all competing nests try to recruit each other. Due to the uniformity of the recruitment process, each competing nest has a constant probability to decrease (or increase) in population. To see why this is true, consider an extreme case where there are two competing nests, one with $n-1$ ants and one with only one ant. By Lemma 3.1.1, the single ant in the small nest has a constant probability of recruiting another ant successfully, which indicates that the small nest increases in population and so the large nest must decrease in population. Similarly, we can show that with constant probability the small nest decreases in population while the large one increases. Following this intuition, in our algorithm, nests with decreasing populations drop out and nests with non-decreasing populations continue competing. Since it is not possible for all nests to experience a decrease at the same time, there is always at least one competing nest. Since each nest has a constant

probability to drop out, similarly to above, we expect a single competing nest to be identified in $\mathcal{O}(\log k)$ steps. This strategy reduces the number of competing nests quickly by depending only on the ants' "luck" during the recruitment process.

Note that the above strategy works well if we consider only ants from competing nests. Otherwise, if we have passive ants participate in the recruitment process together with active ants, it is possible that all competing nests experience increases in population, resulting in a slower drop-out rate. Therefore, in our algorithm, we require all passive ants to not participate in the recruitment process until a single competing nest remains.

Finally, at some point in the execution, only one competing nest remains. However, a number of ants may be passive and not aware that the competition is over. In the final stage of our algorithm, the active ants need to detect that a single competing nest is identified and then recruit all passive ants to the winning nest. To do so, each active ant needs a mechanism to detect that its competing nest is the only competing nest, which indicates that the only ants remaining outside of this competing nest are passive ants. Then, each active ant needs to call the recruit function in the same round as passive ants in order to recruit them to the winning nest. In the worst case, the winning nest has a very small population of active ants, so (following similar reasoning as in the lower bound), it takes $\mathcal{O}(\log n)$ rounds until all passive ants are recruited to the winning nest.

As we will see in the pseudocode, in order to implement the high-level mechanisms mentioned above and achieve proper interleaving of the actions of active and passive ants, we consider phases of the algorithm execution consisting of three rounds each. Roughly speaking, in the first round of each phase, active ants try to recruit each other with the goal of reducing the number of competing nests. In the second round of a phase, active ants that got recruited to a new nest determine whether to be active or passive in the new nest. In the third round of each phase, active ants check whether their nest is the only remaining competing nest. Once a single competing nest is identified, active ants start recruiting in every round, and in this process all passive ants are recruited to the winning nest.

Algorithm 6: Optimal House-Hunting Algorithm

$state$: $\{search, active, passive, final\}$, initially $search$
$nest, nest_r$: nest index $i \in \{0, \cdots, k\}$, initially 0
$count, count_r, count_h, count_n$: integer in $\{0, \cdots, n\}$, initially 0

$(nest, count) :=$ **search**() (R1)
if $q(nest) = 0$ **then**
  $state := passive$
else
  $state := active$

while $state = active$ **do**
  $(nest_r, count_r) :=$ **recruit**(**1**, **nest**) (R1)
  case $nest_r = nest$ *and* $count_r \geq count$ **do**
    go(**nest**) (R2)
    $(\cdot, count_h) :=$ **go**(**0**) (R3)
    if $count_h = count_r$ **then**
      $state := final$
  case $nest_r = nest$ *and* $count_r < count$ **do**
    $state := passive$
    go(**0**) (R2)
    go(**nest**) (R3)
  case $nest_r \neq nest$ **do**
    $(\cdot, count_n) :=$ **go**(**nest**$_\mathbf{r}$) (R2)
    if $count_n < count_r$ **then**
      $state := passive$
    go(**nest**) (R3)
  $nest := nest_r$
  $count := count_r$

while *true* **do**
  case $state = passive$ **do**
    go(**nest**) (R1)
    $(nest_r, \cdot) :=$ **recruit**(**0**, **nest**) (R2)
    if $nest_r \neq nest$ **then**
      $nest := nest_r$
      $state := final$
    go(**nest**) (R3)
  case $state = final$ **do**
    $(nest, \cdot) :=$ **recruit**(**1**, **nest**) (R1, R2, R3)

Detailed Description of the Pseudocode

Each call to the functions from Section 3.1 (in bold) takes exactly one round. The remaining lines of the algorithm are considered to be local computation and are executed in the same round as the preceding function call. Thus, the algorithm matches the structure required by the model.

Consider an ant a that executes Algorithm 6. In the first round, the ant searches randomly for a nest. If the nest has quality 0, the ant moves to the *passive* state; otherwise, it moves to the *active* state. This search is executed only once.

The rest of the code consists of two while loops, each of which takes exactly three rounds to complete an iteration; each round is labeled as either R1, R2, or R3 in the pseudocode. The actions in the three rounds in each subroutine are carefully interleaved in such a way that ants in the active and passive states do not call **recruit**() in the same round until the end of the competition process when a single winning nest remains. To ensure that the loop iterations of active and passive ants take the same number of rounds (three rounds each) and interleave properly, we insert padding **go**$(\cdot)$ calls (these are the calls to **go**$(\cdot)$ in the algorithm in which the output is not assigned to any variable). These function calls also ensure that active and passive ants are never located in the same nests until the last stage of the algorithm when a single competing nest remains.

The first while loop is executed only by active ants. An active ant a tries to recruit other active ants to its competing nest by executing **recruit**$(\mathbf{1}, \mathbf{nest})$. Based on the resulting nest ($nest_r$) and count ($count_r$) values ("r" stands for "after recruitment"), we consider three cases:

- *Case 1:* $nest_r$ is the same as ant a's competing nest and the number of ants in that nest has not decreased. In this case, the nest remains competing. As a result, ant a updates the new count and spends an extra round at the nest that has a special purpose with respect to Cases 2 and 3 below. Finally, ant a checks if the number ($count_h$, "h" stands for "home") of ants at the home nest is the same as the number of ants at its competing nest; if this is the case, it means

that all ants from competing nests have been recruited to a single winning nest and ant a switches to the $final$ state.

- *Case 2:* $nest_r$ is the same as ant a's competing nest but the number of ants has decreased. In this case, the nest drops out. Ant a sets its state to *passive* and spends a round at the home nest, which coincides with the round an active ant spends at its competing nest in Case 1.

- *Case 3:* $nest_r$ is different from ant a's competing nest. This indicates that the ant got recruited to another nest. Although it already knows the number of ants ($count_r$) at the new nest, ant a updates that count ($count_n$, "n" stands for "new count"). The reason for this is to determine whether this new nest is about to compete or drop out. If $count_r = count_n$, the nest is competing because the active ants in Case 1 are spending the same round at the competing nest; if $count_r > count_n$, the nest is dropping out because the ants in Case 2 already determined a decrease in the number of ants and are spending this round at the home nest.

The second while loop is executed only by ants in the passive and final states. We consider two cases based on ant a's state.

- **Passive:** Ant a spends a round at its (non-competing) nest, then it tries to get recruited. This call to **recruit**$(\mathbf{0}, \mathbf{nest})$ never coincides with a **recruit**$(\mathbf{1}, \mathbf{nest})$ of an active ant, so a passive ant can only get recruited by an ant in the $final$ state calling **recruit**$(\mathbf{1}, \mathbf{nest})$. Once successfully recruited, ant a moves to the $final$ state and commits to the winning nest.

- **Final:** Ant a is aware that a single competing nest remains, so it recruits to it in every round. This call to **recruit**$(\mathbf{1}, \mathbf{nest})$ coincides with the call to **recruit**$(\mathbf{0}, \mathbf{nest})$ of passive ants, so once a single nest remains, passive ants are recruited to it in every third round.

Figure 3-1 illustrated the state transitions of Algorithm 6.

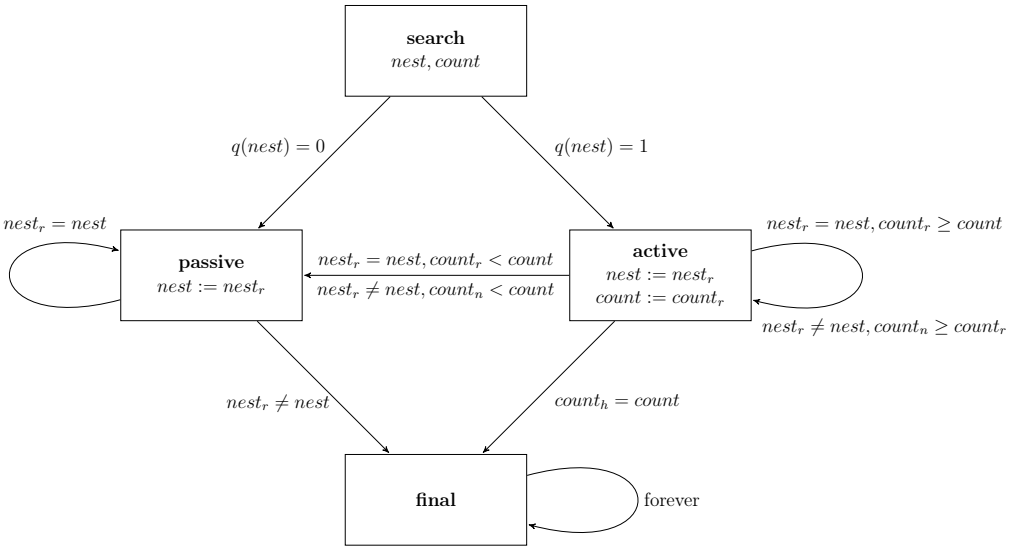

Figure 3-1: State diagram illustration of Algorithm 6. The states in the diagram denote the four possible states of an ant in the algorithm. The variables in the diagram are the following. For any ant, *nest* and *count* are the nest id and population of the current nest of an ant. For an active ant, $nest_r$ and $count_r$ are the nest id and population of the nest after executing **recruit**$(\mathbf{1}, \mathbf{nest})$ in R1, $count_n$ is the population of the new nest an ant got recruited to in R2, and $count_h$ is the population at the home nest in R3. For a passive ant, $nest_r$ is the nest id of the nest after executing **recruit**$(\mathbf{0}, \mathbf{nest})$.

3.3.2 Correctness Proof and Time Bound

As written, Algorithm 6 never terminates; after all ants are committed to the same nest and in the $final$ state, they continue to recruit in every round. This issue can easily be handled if ants check whether the number of ants at the home nest is the same as the number of ants at the competing nest. However, for simplicity, we choose not to complicate the pseudocode and consider the algorithm to terminate once all ants have reached the $final$ state and, thus, committed to the same unique nest.

Proof Overview: The correctness proof and time bound of Algorithm 6 are structured as follows. By defining a slightly different but equivalent recruitment process, we show, in Lemma 3.3.1, that a competing nest is equally likely to continue competing as it is to drop out. Consequently, as we show in Lemma 3.3.2 and Corollary 3.3.3, each competing nest has at least a constant probability of dropping out, indi-

cating that the expected number of competing nests decreases by a constant fraction (Lemma 3.3.4). We put these lemmas together in Theorem 3.3.5 to show that, with high probability, Algorithm 6 solves the HOUSEHUNTING problem in $\mathcal{O}(\log n)$ rounds: $\mathcal{O}(\log k)$ rounds to converge to a single nest and $\mathcal{O}(\log n)$ rounds until all passive ants are recruited to it.

Let R_2 be the set of round numbers r such that $r \equiv 2 \pmod 3$. By the pseudocode, these rounds are exactly the rounds in which active ants recruit other active ants[2].

We define a nest to be *competing* at time r if there is at least one ant in the nest at time r and the ants located in the nest are active at time r (By the pseudocode, and since we assume a synchronous execution, at any given time the ants in a given nest are either all active or all passive). Let $K(r)$ denote the set of competing nests at time r.

For each nest n_i and each time r, let $C(i, r) = \{a \mid \ell(a, r) = i\}$ denote the set of ants located in nest n_i at time r. Let $C(r)$ be the union of all $C(i, r)$ where $n_i \in K(r)$ (n_i is a competing nest at time r). For each nest $n_i \in K(r)$ such that $r + 1 \in R_2$, each ant $a \in C(i, r)$ calls **recruit**$(\mathbf{1}, \mathbf{i})$ in round $r + 1$. That is, for each competing nest n_i, $C(i, r)$ is the set of active ants that recruit to nest n_i, and $C(r)$ is the set of all active ants.

Consider a fixed execution of Algorithm 6 and an arbitrary fixed time $r-1$ in that execution such that $r \in R_2$. We consider the state variables at time $r - 1$ to be fixed, and we consider the probability distribution induced by the random permutation P and the recruitment choices of the ants in round r.

For each ant a, let random variable X_r^a take on values -1, 0 or 1 as follows. If ant a gets recruited by another ant in round r, then $X_r^a = -1$; if ant a successfully recruits another ant, then $X_r^a = 1$; otherwise, $X_r^a = 0$. In the special case where an ant a recruits itself, we define X_r^a to be 0.

For each nest n_i, let random variable Y_r^i denote the net change in the number of ants at nest n_i after recruiting in round r: $Y_r^i = \sum_{a \in C(i,r-1)} X_r^a$. That is, an ant

[2]Note that the rounds in R_2 do not correspond to the rounds labeled R2 in the pseudocode; actually, the rounds in R_2 correspond to the rounds labeled R1. To avoid any confusion, the rest of the section does not refer to the labels of the rounds in the pseudocode.

that recruits successfully contributes a net change of 1 (one new ant) to the nest's population, an ant that is recruited away contributes a net change of -1 to the nest's population, and an ant that is neither recruited away nor recruits successfully, does not contribute to the net change in the nest's population.

Informally speaking, random variable Y_r^i (the change in population of nest n_i) is simply the sum of identically distributed $\{-1, 0, 1\}$ random variables that take on non-zero values with constant probability, and the sum of these variables is negative with constant probability. However, proving this fact requires a more rigorous argument because the X_r^a variables are not independent. We define a slightly different but equivalent recruitment process that we use in the proof of Lemma 3.3.1.

Consider a random variable V that generates a vector v of length $|C(r-1)|$, where initially $v(j) = 0$ for each $j \in [1, |C(r-1)|]$. The values in the vector are updated as follows: at each position $j \in [1, |C(r-1)|]$, starting at position 1 and continuing in order, we choose a uniformly random integer j' between 1 and $|C(r-1)|$. If $j \neq j'$, $v(j) = 0$, and $v(j') = 0$, we set $v(j) := 1$ and $v(j') := -1$. This process is similar to the random choices of the ants in the recruitment process in round r.

Let P be a random variable that generates a uniformly random permutation $p : C(r-1) \to [1, |C(r-1)|]$; that is P assigns to each ant in $C(r-1)$ a random integer between 1 and $|C(r-1)|$. Based on the definition of the random variables V and P, for each ant $a \in C(r-1)$, the random variables $V(P(a))$ and X_r^a are distributed identically[3]. This is true because the two random components that determine the value of X_r^a in the recruitment process (the random permutation and the random choices of the ants) are independent from each other, so we can consider them in either order. In particular, the alternative recruitment process described above fixes some random "recruitment" choices first, and then assigns ants randomly to these choices. In contrast, the original recruitment process in Algorithm 5 fixes a random ordering of the ants first and then assigns random choices to the ants based on this ordering. Both of these recruitment strategies result in the same probability distribution.

[3]This definition of P (mapping from ants to integers) differs slightly from the random permutation (mapping from integers to ants) used in Algorithm 5; however, in both definitions, the functions are simply bijections resulting in equivalent pairing between ants and integers.

First, we show that any competing nest is equally likely to experience an increase or a decrease in population. Based on the pseudocode, this implies that each competing nest is equally likely to continue competing or to drop out.

Lemma 3.3.1. *For each nest* $n_i \in K(r-1)$, $\Pr[Y_r^i < 0] = \Pr[Y_r^i > 0]$.

Proof. Fix a nest $n_i \in K(r-1)$ and consider a fixed vector v generated by random variable V. Since v has an equal number of 1's and -1's (by construction), for each integer $m \in [0, |C(i, r-1)|]$, the number of choices of $|C(i, r-1)|$ indices in v in which the values of v sum up to m is equal to the number of choices in which the values sum up to $-m$. Therefore, since P generates a uniform random permutation, random variables $\sum_{a \in C(i,r-1)} v(P(a))$ and $-\sum_{a \in C(i,r-1)} v(P(a))$ are distributed identically. By the law of total probability, random variables $\sum_{a \in C(i,r-1)} V(P(a))$ and $-\sum_{a \in C(i,r-1)} V(P(a))$ are also distributed identically. We know that the two recruitment processes result in the same distribution, so it follows that Y_r^i and $-Y_r^i$ are distributed identically, implying that $\Pr[Y_r^i < 0] = \Pr[Y_r^i > 0]$. □

Next, we bound the probability that the population of a competing nest remains the same after one round of recruiting, assuming it is not the only competing nest.

Lemma 3.3.2. *For each nest* $n_i \in K(r-1)$, *if* $|K(r-1)| > 1$, *then* $\Pr[Y_r^i = 0] \leq 1535/1536$.

Proof. Fix some nest $n_i \in K(r-1)$. Let random variable $L = |\{a \mid a \in C(i, r-1), X_r^a \neq 0\}|$ and random variable $M = |\{a \mid a \in C(r-1), X_r^a \neq 0\}|$. That is, L denotes the number of ants in $C(i, r-1)$ with non-zero X_r^a variables, and M denotes the number of ants in $C(r-1)$ with non-zero X_r^a variables.

Condition on $L = 2\ell$ and $M = 2m$ for some even integers $2\ell, 2m \in [0, \cdots, |C(r-1)|]$ and assume that $\ell \neq 0$ and $\ell < m$ (we combine all possible cases using the law of total probability at the end of the proof). The only way Y_r^i can be 0 is if the ants from nest n_i have exactly ℓ (out of all m) X_r^a variables equal to 1's, and exactly ℓ (out of all m) X_r^a variables equal to -1's.

$$\Pr[Y_r^i = 0] \leq \frac{\binom{m}{\ell}^2}{\binom{2m}{2\ell}} = \frac{\left(\frac{m!}{(m-\ell)!\ell!}\right)^2}{\frac{(2m)!}{(2m-2\ell)!(2\ell)!}} = \frac{\left(\frac{m(m-1)\cdots(m-\ell+1)}{\ell(\ell-1)\cdots 1}\right)^2}{\frac{2m(2m-1)\cdots(2m-2\ell+1)}{2\ell(2\ell-1)\cdots 1}}.$$

Consider the above expression broken down into factors, each of which consisting of one term from the numerator and two terms from the denominator, as follows:

$$\begin{aligned}\frac{\binom{m}{\ell}^2}{\binom{2m}{2\ell}} &= \left(\frac{\frac{m^2}{\ell^2}}{\frac{2m(2m-1)}{2\ell(2\ell-1)}}\right)\left(\frac{\frac{(m-1)^2}{(\ell-1)^2}}{\frac{(2m-2)(2m-3)}{(2\ell-2)(2\ell-3)}}\right)\cdots\left(\frac{\frac{(m-\ell+1)^2}{1}}{\frac{(2m-2\ell+2)(2m-2\ell+1)}{2\cdot 1}}\right) \\ &= \left(\frac{\frac{m}{\ell}}{\frac{2m-1}{2\ell-1}}\right)\left(\frac{\frac{m-1}{\ell-1}}{\frac{2m-3}{2\ell-3}}\right)\cdots\left(\frac{\frac{m-\ell+1}{1}}{\frac{2m-2\ell+1}{1}}\right).\end{aligned}$$

For each of the terms above, we can verify that, since $\ell < m$, the denominator is at least as large as the numerator, so each term is upper-bounded by 1. Therefore, focusing only on the last term, whose value is maximized for $m - \ell = 1$, we have:

$$\Pr\left[Y_r^a = 0\right] = \frac{\binom{m}{\ell}^2}{\binom{2m}{2\ell}} \leq \frac{m-\ell+1}{2m-2\ell+1} \leq \frac{2}{3}. \tag{3.1}$$

Finally, by the law of total probability:

$$\begin{aligned}\Pr\left[Y_r^i = 0\right] &= \sum_{\ell=0}^{|C(r-1)|}\sum_{m=0}^{|C(r-1)|} \Pr\left[Y_r^i = 0 \mid L = \ell, M = m\right] \cdot \Pr\left[L = \ell, M = m\right] \\ &\leq \sum_{\substack{\ell,m\in[0,|C(i,r-1)|]\\ \ell\neq 0,\ell\neq m,\text{ even }\ell,m}} \Pr\left[Y_r^i = 0 \mid L = \ell, M = m\right] \cdot \Pr\left[L = \ell, M = m\right] \\ &\quad + \Pr\left[Y_r^i = 0 \mid M = L \text{ or } L = 0\right] \cdot \Pr\left[M = L \text{ or } L = 0\right].\end{aligned}$$

Next, we use Equation 3.1, and we consider the remaining values of L and M. Note that $L = 0$ implies that $Y_r^i = 0$, and similarly, $L = M$ implies that $Y_r^i = 0$. Also, since the number of -1's and 1's associated with the X_r^a values of ants in $C(r-1)$ is the same, M must be even. Finally, L being an odd integer implies that $Y_r^i \neq 0$.

$$\begin{aligned}
\Pr\left[Y_r^i = 0\right] &\leq \left(\frac{2}{3}\right) \sum_{\substack{\ell,m \in [0,|C(i,r-1)|] \\ \ell \neq 0, \ell \neq m, \text{ even } \ell,m}} \Pr\left[L = \ell, M = m\right] + \Pr\left[M = L \text{ or } L = 0\right] \\
&= \left(\frac{2}{3}\right)\left(1 - \Pr\left[M = L \text{ or } L = 0\right]\right) + \Pr\left[M = L \text{ or } L = 0\right] \\
&= \left(\frac{2}{3}\right) + \left(\frac{1}{3}\right)\Pr\left[M = L \text{ or } L = 0\right].
\end{aligned}$$

By the assumptions of the lemma that $n_i \in K(r-1)$ and $|K(r-1)| > 1$, it follows that there must be two fixed ants $a, a' \in C(r-1)$ such that $a \in C(i, r-1)$ and $a' \notin C(i, r-1)$. Note that $X_r^a \neq 0$ implies $L \neq 0$, and $X_r^{a'} \neq 0$ implies $M \neq L$. So, $\Pr\left[M = L \text{ or } L = 0\right] \leq 1 - \Pr\left[X_r^a \neq 0 \text{ and } X_r^{a'} \neq 0\right]$. Reasoning similarly to Lemma 3.1.1, we can bound $\Pr\left[X_r^a \neq 0 \text{ and } X_r^{a'} \neq 0\right] \geq 1/512$.

Plugging into the expression for $\Pr\left[Y_r^i = 0\right]$, we get $\Pr\left[Y_r^i = 0\right] \leq 1535/1536$. □

Next, we show that if a nest is not the only competing nest, then it has a constant probability of experiencing a decrease in population.

Corollary 3.3.3. *For each nest $n_i \in K(r-1)$, if $|K(r-1)| > 1$, then $\Pr\left[Y_r^i < 0\right] \geq 1/3072$.*

Proof. By Lemmas 3.3.1 and 3.3.2:

$$\begin{aligned}
\Pr\left[Y_r^i < 0\right] &= 1 - \Pr\left[Y_r^i > 0\right] - \Pr\left[Y_r^i = 0\right] \\
&= 1 - \Pr\left[Y_r^i < 0\right] - \Pr\left[Y_r^i = 0\right] && \text{by Lemma 3.3.1} \\
&\geq \left(\frac{1}{2}\right)\left(1 - \Pr\left[Y_r^i = 0\right]\right) \\
&\geq \frac{1}{3072} && \text{by Lemma 3.3.2.}
\end{aligned}$$

□

Next, we show that the number of competing nests decreases by a constant fraction in expectation after each 3-round iteration of the algorithm.

Lemma 3.3.4. *If* $|K(1)| \geq 1$*, then* $\mathbb{E}[|K(r+2)|] \leq (3071/3072)|K(r-1)|$.

Proof. By Corollary 3.3.3, for each nest $n_i \in K(r-1)$, $\Pr[Y_r^i < 0 \mid |K(r-1)| > 1] \geq 1/3072$. By the pseudocode, the number of competing nests does not change in rounds $r+1$ and $r+2$, so $\mathbb{E}[|K(r+2)| \mid |K(r-1)| > 1] \leq (3071/3072)|K(r-1)|$.

Note that $|K(1)| \geq 1$ implies that at least one good nest is found after the initial round of searching. In subsequent rounds, it is not possible for all competing nests to experience a decrease in population, so for each $r \geq 2$, it is true that $|K(r-1)| \geq 1$. Therefore:

$$\begin{aligned}\mathbb{E}[|K(r+2)|] = \mathbb{E}[|K(r+2)| \mid |K(r-1)| > 1] \cdot \Pr[|K(r-1)| > 1] \\ + \mathbb{E}[|K(r+2)| \mid |K(r-1)| = 1] \cdot \Pr[|K(r-1)| = 1].\end{aligned}$$

Note that $\mathbb{E}[|K(r+2)| \mid |K(r-1)| = 1] = 0$ because once we have a single competing nest at time $r-1$, that nest transitions to the final state in the next round, resulting in 0 competing nests at time $r+2$. So, we have:

$$\begin{aligned}\mathbb{E}[|K(r+2)|] &= \mathbb{E}[|K(r+2)| \mid |K(r-1)| > 1] \cdot \Pr[|K(r-1)| > 1] \\ &\leq \left(\frac{3071}{3072}\right)|K(r-1)|.\end{aligned}$$

□

Finally, we analyze the total running time of Algorithm 6 for an arbitrary probabilistic execution.

Theorem 3.3.5. *For any constant* $c > 1$*, with probability at least* $1 - 1/n^c$*, Algorithm 6 solves the* HouseHunting *problem in* $\mathcal{O}(\log n)$ *rounds.*

Proof. In the first round, all ants search for nests, so the expected number of ants located at each good nest is n/k. Assuming $k \leq n/(12(c+1)\log n) = \mathcal{O}(n/\log n)$, by a Chernoff bound, it follows that, with probability at least $1 - 1/n^{c+1}$, at least $n/(2k)$ ants visits each good nest. Therefore, $\Pr[|K(1)| \geq 1] \geq 1 - 1/n^{c+1}$.

By Lemma 3.3.4, $\mathbb{E}[|K(r+2)| \mid |K(1)| \geq 1] \leq (3071/3072)|K(r-1)|$ for each round $r \in R_2$. Since $|K(r-1)| \leq k$ for each round $r \in R_2$, by the law of iterated

expectation, for $r = \log_{1.0003} k + (c+1)\log_{1.0003} n = \mathcal{O}(\log n)$, it follows that:

$$\mathbb{E}\left[|K(r)| \mid |K(1)| \geq 1\right] \leq 1/n^{c+1}.$$

By a Markov bound, $\Pr\left[|K(r)| \geq 1 \mid |K(1)| \geq 1\right] \leq 1/n^{c+1}$. In other words, after $\mathcal{O}(\log n)$ rounds, with probability at least $1 - 1/n^{c+1}$, there are no competing nests. Conditioning on at least one good nest being found after the first round of searching, and using the fact that it is not possible for all good nests to drop out in any given round, there is at least one nest either competing or in the final state after $\mathcal{O}(\log n)$ rounds. Therefore, conditioning on at least one good nest being found after the first round of searching, after $\mathcal{O}(\log n)$ rounds, with probability at least $1 - 1/n^{c+1}$, there is *exactly* one nest in the final state. Union-bounding over the first round of searching and the subsequent competition rounds, with probability at least $1 - 1/n^c$, there is a single nest in the final state after $\mathcal{O}(\log n)$ rounds.

After there is exactly one nest in the final state, by the pseudocode, all ants committed to it start recruiting the passive ants during every third round. Each recruited ant also transitions to the final state, resulting in at most $\mathcal{O}(\log n)$ rounds until all ants are committed to the winning nest. □

3.4 Simple Algorithm

We now give a very simple algorithm (Algorithm 7) that solves the HouseHunting problem in $\mathcal{O}(k^3 \log^{1.5} n)$ rounds with high probability. In contrast to Algorithm 6, the algorithm in this section is much simpler and capable of using only approximate information from the environment to solve the house hunting problem (which we will see in Section 3.5). The main idea of the algorithm is that all ants initially search for nests and those that find good nests simply continuously recruit to their nests with probability proportional to nest population in each round. Ants in larger nests recruit at higher rates, and eventually their populations swamp the populations of smaller nests. This process is similar to the well-known *Polya's urn* model [27].

3.4.1 Algorithm Pseudocode and Description

In the first round of Algorithm 7, all ants search for nests; this search is executed only once, in the first round of the algorithm execution. The algorithm proceeds in subsequent rounds of recruitment by all ants, where each ant is either actively recruiting by calling **recruit**$(\mathbf{1}, \cdot)$, or trying to get recruited by calling **recruit**$(\mathbf{0}, \cdot)$. In each round, each ant chooses whether to recruit or not with probability $(count \cdot q(nest))/n$, where $count$ and $q(nest)$ are the assessed population and quality of the nest, and n is the total number of ants. Recall that we assume the quality of a nest is a binary value in $\{0, 1\}$, so if a nest is not good, ants simply do not recruit to it. Each iteration of the while loop takes exactly one round, which consist of a call to **recruit**$(\cdot, \mathbf{nest})$ by each ant.

Algorithm 7: Simple House-Hunting Algorithm

$nest$: nest index in $\{0, \cdots, k\}$, initially 0
$count$: integer in $\{0, \cdots, n\}$, initially 0
b : binary value in $\{0, 1\}$, initially 0
$(nest, count) :=$ **search**()
while *true* **do**
  $b := 1$ with probability $(q(nest) \cdot count)/n$, 0 otherwise
  $(nest, count) :=$ **recruit**$(\mathbf{b}, \mathbf{nest})$

3.4.2 Main Result and Proof Overview

We will prove the following main result about the correctness and running time of Algorithm 7.

Theorem 3.4.1. *For any constant $c > 1$, with probability at least $1 - 1/n^c$, Algorithm 7 solves the* HouseHunting *problem in $2^{22}(c+7)(c+3)k^3 \log^{1.5} n = \mathcal{O}(k^3 \log^{1.5} n)$ rounds.*

Proof Overview: The proof consists of three main parts. First, in Section 3.4.3, we introduce a few properties of the recruitment step of Algorithm 7: in Lemmas 3.4.2, 3.4.3, 3.4.4, and 3.4.5, we analyze the expected change of the population of a

single nest after a single round of recruiting. The remainder of the proof is broken down into two parts that correspond to the two main stages of the execution of the algorithm: the symmetry-breaking stage (Section 3.4.4) and the drop-out stage (Section 3.4.5). In the symmetry-breaking stage, we consider a fixed number of rounds (in $\mathcal{O}(k^3 \log^{1.5} n)$) and we show that within that number of rounds we can break the initial symmetry between each pair of the nests (that is, the population gap between the nests is large). In Lemma 3.4.6, we compute the expected growth of the population gap between any two nests in a single round. Then, in Lemma 3.4.15, we use a general result about Markov chains (Lemma 3.4.14) to show that in $\mathcal{O}(k^3 \log^{1.5} n)$ rounds the gap between any two nests is fairly large with high probability. After the symmetry-breaking stage, we enter the drop-out stage, which also consists of a fixed number of rounds (in $\mathcal{O}(k \log n)$), and we show that at least one nest of each pair of nests drops out completely within that number of rounds. In particular, in Lemmas 3.4.17 and 3.4.18 we show that once a nest drops below a certain threshold, it decreases in population fairly quickly with high probability. Finally, we bring all results together by concluding (in Lemma 3.4.19 and Theorem 3.4.1) that, with high probability, at the end of the drop-out stage, at least one nest from each pair of nests has no ants. Union-bounding over all possible pairs of nests, we show that, with high probability, all the ants are located in a single nest.

For each nest n_i and each time r, let random variable $p(i, r) = c(i, r)/n$ denote the proportion of ants at nest n_i at time r. Define $\Sigma(r) = \sum_{i=1}^{k} p(i, r)^2$. Since $\sum_{i=1}^{k} p(i, r) = 1$, by the Cauchy-Schwarz inequality $\Sigma(r) \geq 1/k$.

Note that for a fixed time r (and fixed $p(i, r)$ for $i \in [0, k]$), in round $r + 1$ each ant uses the population at time r to determine its recruitment probability. By the pseudocode, ants located at a good nest n_i at time r recruit with probability $p(i, r)$ in round $r + 1$. Also, $\Sigma(r)$ is the expected proportion of ants that call **recruit**$(\mathbf{1}, \cdot)$ in round $r + 1$ (each ant from nest n_i recruits with probability $p(i, r)$, and there is a $p(i, r)$ fraction of all the ants in nest n_i).

Let $c > 1$ be a fixed arbitrary constant, and let $d = 4/3$. For the sake of analysis, assume that $k \leq 64((c + 7)n/\log n)^{1/4}$.

3.4.3 Properties of the Recruitment Process

We first study how the population of a single nest changes in a single round. Intuitively, in round r, we expect a $p(i, r-1)$ fraction of the ants in nest n_i to recruit to nest n_i, and approximately a $\Sigma(r-1)$ fraction of the ants in nest n_i to be recruited away from nest n_i. Therefore, we expect the fraction of ants in nest n_i to change by approximately $p(i, r-1)(p(i, r-1) - \Sigma(r-1))$. We show this in Lemma 3.4.5, modulo constant terms and factors. Before we are ready to prove Lemma 3.4.5, we prove a few results that describe the expected outcome of a single ant recruiting.

For the rest of this section, consider a fixed execution of Algorithm 7 and an arbitrary fixed time $r - 1 > 0$ in that execution. We consider the state variables at time $r - 1$ to be fixed, and we consider the probability distribution induced by choosing the random bit b for each ant, and the recruitment process (the random permutation and the random choices of the ants) in round r.

For each ant a, let random variable X_r^a take on values:

- 1 if a is the first element of a pair in the set of successful recruitments M, except if the pair is (a, a) (that is, a recruits itself),
- -1 if a is the second element of a pair in the set of successful recruitments M, except if the pair is (a, a),
- 0 if a does not appear in any recruitment pair in the set of successful recruitments M, or if $(a, a) \in M$.

Note that X_r^a can be 1 even if an ant recruits one of its own nest mates. The recruited nest mate will have a corresponding value of -1, resulting in no net gain of ants in the nest. When an ant recruits itself we define X_r^a to be 0.

Let random variable $b(a, r) \in \{0, 1\}$ denote the recruitment bit of ant a in round r (denoted as b in the pseudocode), and let $\alpha(a, r) = \Pr[X_r^a = 1 \mid b(a, r) = 1]$ and $\beta(a, r) = \Pr[X_r^a = -1]$. That is, α denotes the probability that an ant succeeds in recruiting another ant assuming the first ant is recruiting, and β denotes the probability that an ant is recruited. Next, we bound the value of $\alpha(a, r)$.

Lemma 3.4.2. *For each pair of ants a and a', $\alpha(a,r) = \alpha(a',r) \geq 1/16$.*

Proof. Fix an arbitrary pair of ants a and a'. For each ant a with $b(a,r) = 1$, the probability it recruits successfully is:

$$\begin{aligned}
\alpha(a,r) &= \Pr\left[X_r^a = 1 \mid b(a,r) = 1\right] \\
&= \sum_{m=1}^{n} (\Pr\left[X_r^a = 1 \mid b(a,r) = 1, a \text{ is at position } m \text{ in } P\right] \\
&\qquad \cdot \Pr\left[a \text{ is at position } m \text{ in } P\right]) \\
&= \left(\frac{1}{n}\right) \sum_{m=1}^{n} \Pr\left[X_r^a = 1 \mid b(a,r) = 1, a \text{ is at position } m \text{ in } P\right] \\
&= \left(\frac{1}{n}\right) \sum_{m=1}^{n} \sum_{a^*=1}^{n} (\Pr\left[X_r^a = 1 \mid b(a,r) = 1, a \text{ is at position } m \text{ in } P, a \text{ recruits } a^*\right] \\
&\qquad\qquad \cdot \Pr\left[a \text{ recruits } a^*\right]) \\
&= \left(\frac{1}{n^2}\right) \sum_{m=1}^{n} \sum_{a^*=1}^{n} \Pr\left[X_r^a = 1 \mid b(a,r) = 1, a \text{ is at position } m \text{ in } P, a \text{ recruits } a^*\right].
\end{aligned}$$

Note that the last expression is the same for ants a and a' because $b(a,r) = b(a',r) = 1$, and the two ants are identical with respect to their positions in P and their random choices. Therefore, $\alpha(a,r) = \alpha(a',r)$. By Lemma 3.4.2, $\alpha(a,r) \geq 1/16$. □

Because of Lemma 3.4.2, we can define $\alpha(r) = \alpha(a,r)$ for all ants a in round r.

Lemma 3.4.3. *For each ant a, $\beta(a,r) = \alpha(r) \cdot \Sigma(r-1) \geq 1/(16k)$.*

Proof. Fix an arbitrary ant a. The probability that ant a is recruited in round r is:

$$\begin{aligned}
\beta(a,r) &= \Pr\left[X_r^a = -1\right] \\
&= \sum_{a'=1}^{n} \Pr\left[a' \text{ successfully recruits } a\right] \\
&= \sum_{i=1}^{k} \sum_{\{a' \mid \ell(a', r-1) = i\}} \Pr\left[a' \text{ successfully recruits } a\right] \\
&= \sum_{i=1}^{k} \sum_{\{a' \mid \ell(a', r-1) = i\}} p(i, r-1)\alpha(r) \left(\frac{1}{n}\right),
\end{aligned}$$

where $p(i, r-1)$ is the probability that ant a' chooses to recruit (calls **recruit**$(\mathbf{1}, \mathbf{i})$), $\alpha(r)$ is the probability that ant a' succeeds in recruiting another ant, and $1/n$ is the probability that ant a' chooses ant a as part of the recruitment process. So, we have:

$$\begin{aligned}\beta(a, r) &= \sum_{i=1}^{k} np(i, r-1)^2 \alpha(r) \left(\frac{1}{n}\right) \\ &= \sum_{i=1}^{k} p(i, r-1)^2 \alpha(r) \\ &= \alpha(r) \cdot \Sigma(r-1).\end{aligned}$$

Finally, by Lemma 3.4.2 and by the fact that $\Sigma(r-1) \geq 1/k$, $\beta(a, r) \geq 1/(16k)$. □

Because of Lemma 3.4.3, we can define $\beta(r) = \beta(a, r)$ for all ants a in round r.

Next, we bound the expected value of the recruitment contribution X_r^a of each ant a to its nest.

Lemma 3.4.4. *For each nest n_i, and each ant a such that $\ell(a, r-1) = i$, $\mathbb{E}[X_r^a] = \alpha(r)(p(i, r-1) - \Sigma(r-1))$.*

Proof. Fix an arbitrary nest n_i, and an ant a such that $\ell(a, r-1) = i$. By definition, $\mathbb{E}[X_r^a] = \Pr[X_r^a = 1] - \Pr[X_r^a = -1]$. By Lemmas 3.4.2 and 3.4.3, $\mathbb{E}[X_r^a] = \alpha(r)p(i, r-1) - \beta(r) = \alpha(r)(p(i, r-1) - \Sigma(r-1))$. □

Next, we use Lemma 3.4.4 to calculate the expected change in the value of $p(i, r)$ after one round of recruiting.

Lemma 3.4.5. *For each nest n_i, $\mathbb{E}[p(i, r)] = p(i, r-1)(1+\alpha(r)(p(i, r-1)-\Sigma(r-1)))$.*

Proof. Fix an arbitrary nest n_i and an arbitrary ant a' in nest n_i at time $r-1$.

By linearity of expectation:

$$\begin{aligned}\mathbb{E}\left[p(i,r)\right] &= p(i,r-1) + \left(\frac{1}{n}\right) \sum_{\{a|\ell(a,r-1)=i\}} \mathbb{E}\left[X_r^a\right] \\ &= p(i,r-1) + \left(\frac{c(i,r-1)}{n}\right) \mathbb{E}\left[X_r^{a'}\right] \\ &= p(i,r-1)(1 + \mathbb{E}\left[X_r^{a'}\right]),\end{aligned}$$

The $(1/n)$ factor in front of the sum converts the expression in the sum from an absolute number of ants to a fraction of ants (as required by the definition of $p(i,r)$). In the second equality, a' is a fixed ant in nest n_i and the equality follows from the fact that X_r^a is identically distributed for each ant in the same nest.

By Lemma 3.4.4, for ant a' in nest n_i, $\mathbb{E}\left[X_r^{a'}\right] = \alpha(r)(p(i,r-1) - \Sigma(r-1))$, so:

$$\mathbb{E}\left[p(i,r)\right] = p(i,r-1)(1 + \alpha(r)(p(i,r-1) - \Sigma(r-1))).$$

□

3.4.4 Symmetry Breaking between Two Nests

In this section, we focus on the relative gap between the populations of any two nests. First, in Lemma 3.4.6 we bound the expected increase in this gap after one round of recruiting. Then, we define a new random variable that helps us analyze the growth in the gap and prove some properties of this random variable in Lemmas 3.4.9 and 3.4.11 (and Lemmas 3.4.12 and 3.4.13). Finally, in Lemma 3.4.15, we show that, with high probability, in $\mathcal{O}(k^3 \log^{1.5} n)$ rounds, the gap between the populations of the nests grows sufficiently. The remaining results in this section (Lemmas 3.4.10 and 3.4.14, and Claims 3.4.7 and 3.4.8) are auxiliary results.

For the following results (Lemmas 3.4.6, 3.4.9, 3.4.10, 3.4.11), consider a fixed execution of Algorithm 7 and an arbitrary fixed time $r-1 > 0$ in that execution. We consider the state variables at time $r-1$ to be fixed, and we consider the probability distribution over the randomness in round r.

Recall that $d = 4/3$.

Lemma 3.4.6. *Let n_i and n_j be a pair of nests such that $p(i, r-1), p(j, r-1) \geq \Sigma(r-1)/d$. Then, $\mathbb{E}\left[|p(i,r) - p(j,r)|\right] \geq |p(i,r-1) - p(j,r-1)|(1 + \Sigma(r-1)/32)$.*

Proof. Fix an arbitrary pair of nests n_i and n_j such that $p(i, r-1), p(j, r-1) \geq \Sigma(r-1)/d$. Without loss of generality, assume $p(i, r-1) \geq p(j, r-1)$. Then, by Lemma 3.4.5, it follows that:

$$\begin{aligned}
&\mathbb{E}\left[p(i,r) - p(j,r)\right] \\
&\quad = p(i,r-1)(1 + \alpha(r)(p(i,r-1) - \Sigma(r-1))) \\
&\quad - p(j,r-1)(1 + \alpha(r)(p(j,r-1) - \Sigma(r-1))) \\
&\quad = (p(i,r-1) - p(j,r-1))(1 + \alpha(r)(p(i,r-1) + p(j,r-1) - \Sigma(r-1))).
\end{aligned}$$

By the assumption that $p(i, r-1), p(j, r-1) \geq \Sigma(r-1)/d$, it follows that:

$$\begin{aligned}
&\mathbb{E}\left[|p(i,r) - p(j,r)|\right] \\
&\quad \geq \mathbb{E}\left[p(i,r) - p(j,r)\right] \\
&\quad \geq (p(i,r-1) - p(j,r-1)) \left(1 + \alpha(r) \left(\frac{2\Sigma(r-1)}{d} - \Sigma(r-1)\right)\right) \\
&\quad \geq (p(i,r-1) - p(j,r-1)) \left(1 + \left(\frac{\Sigma(r-1)}{16}\right) \left(\frac{2}{d} - 1\right)\right) \\
&\quad = (p(i,r-1) - p(j,r-1)) \left(1 + \frac{\Sigma(r-1)}{32}\right) \qquad \text{since } d = \frac{4}{3},
\end{aligned}$$

where the second inequality follows from Lemma 3.4.2. □

Next, we analyze how the gap between any two nests increases for all possible values of the populations of the nests (not just when they are at least $\Sigma(r-1)/d$), and also, how long it takes for the gap to grow sufficiently. In particular, let $m = 128\sqrt{(c+7)\log n/n}$; we are interested in growing the gap to at least m.

The following random variables $I(i,j,r)$ and $Y(i,j,r)$ are defined with respect to a pair of nests n_i and n_j and the probability distribution over all possible executions of Algorithm 7. Both random variables are derived from the random variables $p(i,r)$ for

$i \in \{1, \cdots, k\}$. Recall that the variables $p(i, r)$, for $i \in \{1, \cdots, k\}$ (and consequently the derived variable $\Sigma(r)$) depend only on the state at the time $r - 1$.

For each time r and each pair of nests n_i and n_j, let indicator random variable $I(i, j, r)$ be defined as follows:

$$I(i,j,r) = \begin{cases} 1, & \text{if } \min\{p(i,r), p(j,r)\} < \Sigma(r)/d, \\ 0, & \text{otherwise.} \end{cases}$$

Random variable $I(i, j, r)$ captures whether or not at least one of the nests n_i or n_j has a population less than $\Sigma(r)/d$. If this is the case, then we will show that, in the drop-out stage (Section 3.4.5), the small nest drops out fairly quickly. It could be the case that both n_i and n_j are small, in which case they both drop out quickly.

Also, let random variable $Y(i, j, r)$ be defined as follows:

$$Y(i,j,r) = |p(i,r) - p(j,r)| + m \cdot I(i,j,r).$$

We use random variable $Y(i, j, r)$ to represent the gap between the two nests while both nests are fairly large. When at least one nest drops below the $\Sigma(r)/d$ threshold, the value of $Y(i, j, r)$ defaults to the target value m. The goal is to show that at some point either the population gap becomes at least m, or at least one of the nests has population less than $\Sigma(r)/d$; we show this with high probability in Lemma 3.4.15.

The following simple claim follows by the definition of $Y(i, j, r)$.

Claim 3.4.7. *For each pair of nests n_i and n_j, if $Y(i, j, r) < m$ then $I(i, j, r) = 0$, $p(i, r), p(j, r) \geq \Sigma(r)/d$, and $|p(i, r) - p(j, r)| < m$.*

Next, in Lemma 3.4.9 we will show that, assuming $Y(i, j, r - 1)$ is small (that is, the gap between the nests is less than m and both nests are fairly large), the expected value of the difference between $Y(i, j, r)$ and $Y(i, j, r-1)$ is $\Omega(1/k^3\sqrt{n})$. In the proof, we consider two cases: (1) the gap between the nests at time $r - 1$ is relatively small, and (2) the gap between the nests is fairly large (but still smaller than m). We use some standard results about the binomial distribution, applied to a restricted case of

our recruitment process, in case (1), and Lemma 3.4.6 in case (2), in order to show that in both cases the gap between the nests grows by $\Omega(1/k^3\sqrt{n})$ in expectation.

Before we proceed with the lemma, we define two methods of generating the recruitment pairs in round r. Method 1 is the original recruitment process defined in Section 3.1, and Method 2 is a slight variation of that recruitment process. We will show in Claim 3.4.8 that both methods result in the same distribution of successful recruitment pairs, and hence, the same distribution of executions. Then, we will use Method 2 in Lemma 3.4.9 to lower bound the expected gap between the new populations of nests n_i and n_j, which will also imply a lower bound on the expected gap in the original case (Method 1).

Method 1: Let R be the set of ants that call **recruit**$(\cdot,\cdot)$ in round r (in Algorithm 7, R is always the entire set of ants). Let random variable $B : R \to \{0,1\}$ be a mapping (from ants to booleans) that represents the recruitment bits of the ants in round r. Let random variable $G : R \to R$ be a mapping (from ants to ants) such that $G = g$ uniformly at random among all fixed mappings $g : R \to R$; here, G represents the recruitment choices of the ants in round r. Finally, let random variable $P : R \to [1, |R|]$ be a bijection (from ants to unique integers) such that $P = p$ uniformly at random among all fixed bijections $p : R \to [1, |R|]$; here P represents the random permutation that determines the order in which ants recruit in round r. The random variables B, G, and P are independent from each other, and together they determine the set M of successful recruitment pairs in round r (as shown in Algorithm 5).

Method 2: Consider the triple (G, B, P) defined in Method 1 and the resulting set M of successful recruitment pairs. Let random variable $K = \{(a, a') \mid (a, a') \in M, \ell(a, r-1), \ell(a', r-1) \in \{n_i, n_j\}, \ell(a, r-1) \neq \ell(a', r-1), B(a) = B(a') = 1\}$. That is K is the subset of M that contains all "key" pairs where one ant is from nest n_i, the other ant is from nest n_j, and both ants are actively recruiting in round r. Let random variable $P' : R \to [1, |R|]$ be a bijection (from ants to unique integers) such that for each pair $(a, a') \in K$, independently from all other pairs in K, with probability $1/2$, $P'(a) = P(a')$ and $P'(a') = P(a)$; with the remaining probability $1/2$, $P'(a) = P(a)$

and $P'(a') = P(a')$. For each ant a not involved in a pair in K, $P'(a) = P(a)$. That is, for each pair of ants in $(a, a') \in K$, with probability $1/2$, P' switches the positions of the ants a and a' assigned by P. Similarly to Method 1, we determine the set M' of successful recruitments using the random variables (G, B, P').

Claim 3.4.8. *The set M of recruitment pairs resulting from the triple of random variables (G, B, P) generated by Method 1 is identically distributed to the set M' of recruitment pairs resulting from the triple of random variables (G, B, P') generated by Method 2.*

Proof of Claim. Let g, b, and p_1 be fixed values of the random variables G, B, and P, respectively. Consider an arbitrary fixed recruitment pair (a_1, a_2) in the set of key recruitment pairs generated by (g, b, p_1). Let p_2 be the fixed mapping from ants to integers such that $p_2(a_1) = p_1(a_2)$, $p_2(a_2) = p_1(a_1)$ and $p_2(a') = p_1(a')$ for all $a' \neq a_1$ and $a' \neq a_2$. That is, p_2 is identical to p_1 except for flipping the positions of ants a_1 and a_2. Therefore, since $b(a_1) = b(a_2) = 1$, the pair (a_2, a_1) must be in the set of key recruitment pairs generated by (g, b, p_2).

Consider random variable P' with respect to flipping the positions of ants a_1 and a_2 (and no other pairs of ants). Based on the definitions above, we have that $\Pr[P' = p_1 \mid G = g, B = b, P = p_1] = 1/2$ (resulting from P' *not* switching the positions of a_1 and a_2) and $\Pr[P' = p_1 \mid G = g, B = b, P = p_2] = 1/2$ (resulting from P' switching the positions of a_1 and a_2). By the definition of P, we know that $\Pr[P = p_1 \mid G = g, B = b] = \Pr[P = p_2 \mid G = g, B = b]$. Therefore, by the law of total probability:

$$\begin{aligned}
&\Pr[P' = p_1 \mid G = g, B = b] \\
&\quad = \Pr[P' = p_1 \mid G = g, B = b, P = p_1] \cdot \Pr[P = p_1 \mid G = g, B = b] \\
&\quad + \Pr[P' = p_1 \mid G = g, B = b, P = p_2] \cdot \Pr[P = p_2 \mid G = g, B = b] \\
&\quad = \left(\frac{1}{2}\right) \Pr[P = p_1 \mid G = g, B = b] + \left(\frac{1}{2}\right) \Pr[P = p_2 \mid G = g, B = b] \\
&\quad = \Pr[P = p_1 \mid G = g, B = b]
\end{aligned}$$

So, with respect to a single recruitment pair, P and P' are distributed identically. By definition, P' is formed by a sequence of independent flips of recruitment pairs, so overall, conditioning on $G = g$ and $B = b$, P and P' are distributed identically. By the law of total probability, we have that (G, B, P') and (G, B, P) are distributed identically. The claim follows since the recruitment pairs are given by a deterministic function of these random variables. □

Next, we use Method 2 to lower bound the expected difference between $Y(i, j, r)$ and $Y(i, j, r-1)$. Let $d' = 2^8$.

Lemma 3.4.9. *For each pair of nests n_i and n_j such that $Y(i, j, r-1) < m$, it is true that:*

$$\mathbb{E}\left[Y(i, j, r)\right] \geq Y(i, j, r-1) + \frac{1}{16d'k^3\sqrt{n}}.$$

Proof. Fix an arbitrary pair of nests n_i and n_j such that $Y(i, j, r-1) < m$. By Claim 3.4.7, $p(i, r-1), p(j, r-1) \geq \Sigma(r-1)/d$. We consider two cases based on the value of $Y(i, j, r-1)$.

Case 1: $Y(i, j, r-1) < 2/(d'k^2\sqrt{n})$. In this case, we will actually show a stronger result: $\mathbb{E}\left[Y(i, j, r)\right] \geq Y(i, j, r-1) + 1/(d'k^2\sqrt{n})$.

We show that determining the successful recruitment pairs based on Method 2 results in an expected growth of the gap between nests n_i and n_j. By Claim 3.4.8, the result carries over to the original recruitment process (Method 1) as well.

Let g, b, and p be fixed values of the random variables G, B, and P, respectively. Note that fixing G, B, and P also fixes the set K of key pairs; let k' be that fixed value of K. Random variable P' is the only remaining source of randomness that we analyze next; it determines the order of the ants in each pair in the set k' of key pairs. Therefore, we can examine each fixed recruitment pair not in k' and determine the change in nest populations that this pair causes. That is, for each pair $(a, a') \notin k'$, such that $\ell(a, r-1) = x$ and $\ell(a', r-1) = y$, the population of nest n_x increases by one ant with respect to its value at time $r-1$, and the population of nest n_y decreases by one ant with respect to its value at time $r-1$. Let $p'(i, r)$ and $p'(j, r)$ denote the

populations of nests n_i and n_j, respectively, after evaluating all fixed recruitment pairs not in k'. Without loss of generality, assume $p'(i,r) \geq p'(j,r)$.

Consider the x'th recruitment pair in k' (the ordering of the recruitment pairs is not important here, we use an index x only for identifying the pairs). Let random variable $Z_x = -1$ if the first ant in the pair is located in nest n_j at time $r-1$, and let $Z_x = 1$ if the first ant from the pair is located in nest n_i at time $r-1$. Random variable Z_x represents the order of the ants in the x'th recruitment pair.

Note that, based on the definition of P', each Z_x is identically and independently distributed, taking on values -1 and 1 with probability 1/2 each. The sum Z, of the Z_x random variables is a (shifted) binomial random variable with $|k'|$ trials and success probability 1/2, and it represents the net number of ants from nest n_j that join nest n_i. It is well-known that $|Z| \geq \sqrt{|k'|}$ with constant probability. This can be proved, for example, using the Paley-Zygmunt inequality (Theorem A.1.8 in the Appendix). We have $\mathbb{E}[Z^2] = |k'|$ and $\mathbb{E}[Z^4] = |k'| + 3|k'|(|k'|-1) = 3|k'|^2 - 2|k'|$. So, by the Paley-Zygmunt inequality:

$$\begin{aligned}\Pr\left[|Z| \geq \frac{\sqrt{|k'|}}{2}\right] &= \Pr\left[Z^2 \geq \frac{|k'|}{4}\right] = \Pr\left[Z^2 \geq \frac{\mathbb{E}[Z^2]}{4}\right] \\ &\geq \left(1-\frac{1}{4}\right)^2 \left(\frac{\mathbb{E}[Z^2]^2}{\mathbb{E}[Z^4]}\right) \geq \left(\frac{1}{2}\right)\frac{|k'|^2}{3|k'|^2 - 2|k'|} \geq \frac{1}{6}.\end{aligned}$$

By symmetry, we have that $\Pr\left[Z \geq \sqrt{k'}/2\right] \geq 1/12$. This inequality holds for any fixed triple (g,b,p) of values corresponding to random variables (G,B,P), and consequently, for any resulting value $|k'|$ of random variable $|K|$. By the law of total probability, we have:

$$\Pr\left[Z \geq \frac{\sqrt{|k'|}}{2}\middle| K = k'\right] \geq \frac{1}{12}. \tag{3.2}$$

Next, we bound random variable $|K|$, starting with its expectation. The proba-

bility distribution here is based on all random variables B, G, and P.

$$\begin{aligned}\mathbb{E}\left[|K|\right] &\geq np(i,r-1)^2 p(j,r-1)^2 \alpha(r) \\ &\geq \frac{n}{16d^4k^4} \text{by Lemma 3.4.2 and since } p(i,r-1), p(j,r-1) \geq \frac{\Sigma(r-1)}{d} \geq \frac{1}{dk}.\end{aligned}$$

The success of each recruitment is negatively correlated with the success of the other recruitments, so we can apply a Chernoff bound (see Theorem 4.3 in [46]) to show that $|K|$ concentrates around its expectation. By the assumption that $k \leq 64((c+7)n/\log n)^{1/4}$ we have $\mathbb{E}\left[|K|\right] = \Omega(\log n)$ and hence:

$$\Pr\left[|K| \geq \frac{\mathbb{E}\left[|K|\right]}{2} \geq \frac{n}{32d^4k^4}\right] \geq \frac{1}{2}. \tag{3.3}$$

By Equations (3.2) and (3.3), and the law of total probability, we get:

$$\begin{aligned}\Pr\left[Z \geq \frac{\sqrt{n}}{8\sqrt{2}d^2k^2}\right] &\geq \sum_{|k'|=n/(32d^4k^4)}^{\infty} \Pr\left[Z \geq \frac{\sqrt{|k'|}}{2} \middle| |K| = |k'|\right] \cdot \Pr\left[|K| = |k'|\right] \\ &\geq \left(\frac{1}{12}\right) \sum_{|k'|=n/(32d^4k^4)}^{\infty} \Pr\left[|K| = |k'|\right] \\ &\geq \left(\frac{1}{12}\right) \Pr\left[|K| \geq \frac{n}{32d^4k^4}\right] \geq \frac{1}{24}.\end{aligned}$$

Recall that $p'(i,r) \geq p'(j,r)$ and $p(\cdot,\cdot) = c(\cdot,\cdot)/n$.

$$\begin{aligned}\Pr\left[|p(i,r) - p(j,r)| \geq \frac{1}{8\sqrt{2}d^2k^2\sqrt{n}}\right] &\geq \Pr\left[p'(i,r) - p'(j,r)) + \frac{Z}{n} \geq \frac{1}{8\sqrt{2}d^2k^2\sqrt{n}}\right] \\ &\geq \Pr\left[\frac{Z}{n} \geq \frac{1}{8\sqrt{2}d^2k^2\sqrt{n}}\right] \geq \frac{1}{24}.\end{aligned}$$

By the law of total expectation and by the assumption that $d' = 2^8$, we have:

$$\mathbb{E}\left[|p(i,r) - p(j,r)|\right] \geq \frac{1}{192\sqrt{2}d^2k^2\sqrt{n}} \geq \frac{3}{d'k^2\sqrt{n}}.$$

By Claim 3.4.8, the recruitment pairs generated by Method 2 are distributed identically as the recruitment pairs generated by Method 1. Therefore, the expected

difference between the populations of nests n_i and n_j is also distributed identically in the two methods. So, in Method 1 we have:

$$\mathbb{E}\left[Y(i,j,r)\right] \geq \mathbb{E}\left[|p(i,r) - p(j,r)|\right] \geq \frac{3}{d'k^2\sqrt{n}}.$$

By the assumption that $Y(i,j,r-1) < 2/(d'k^2\sqrt{n})$, we have that $\mathbb{E}\left[Y(i,j,r)\right] \geq Y(i,j,r-1) + 1/(d'k^2\sqrt{n})$.

Case 2: $Y(i,j,r-1) \geq 2/(d'k^2\sqrt{n})$. In this case, since $I(i,j,r-1) = 0$, it must be the case that $|p(i,r-1) - p(j,r-1)| \geq 2/(d'k^2\sqrt{n})$. By the assumption that $p(i,r-1), p(j,r-1) \geq \Sigma(r-1)/d$, we can apply Lemma 3.4.6. Recall that by the Cauchy-Schwarz inequality $\Sigma(r-1) \geq 1/k$.

$$\begin{aligned}
\mathbb{E}\left[|p(i,r) - p(j,r)|\right] &\geq |p(i,r-1) - p(j,r-1)| \left(1 + \frac{\Sigma(r-1)}{32}\right) \\
&\geq |p(i,r-1) - p(j,r-1)| + \left(\frac{2}{d'k^2\sqrt{n}}\right)\left(\frac{1}{32k}\right) \\
&\geq |p(i,r-1) - p(j,r-1)| + \frac{1}{16d'k^3\sqrt{n}}.
\end{aligned}$$

By the definition of $Y(i,j,r)$:

$$\begin{aligned}
\mathbb{E}\left[Y(i,j,r)\right] &\geq \mathbb{E}\left[|p(i,r) - p(j,r)|\right] \\
&\geq |p(i,r-1) - p(j,r-1)| + \frac{1}{16d'k^3\sqrt{n}} \\
&= Y(i,j,r-1) + \frac{1}{16d'k^3\sqrt{n}} \qquad \text{since } I(i,j,r-1) = 0.
\end{aligned}$$

□

Next, we prove a helper lemma that lets us show that the population of any given large nest (with population proportion larger than $\Sigma(r-1)/d$) concentrates around its expectation.

Lemma 3.4.10. *For each nest n_i such that $p(i,r-1) \geq \Sigma(r-1)/d$,*
$\Pr\left[|p(i,r) - \mathbb{E}\left[p(i,r)\right]| \leq m/(128k)\right] \geq 1 - 1/n^{c+6}$.

Proof. Consider the successful recruitment pairs in round r that involve ants from nest n_i. The recruitment pairs that involve two ants from nest n_i do not result in any change in the population of nest n_i, so we ignore them for the purposes of this bound. Consider the remaining two types of recruitment pairs: (1) the pairs in which the first ant is from nest n_i, and (2) the pairs in which the second ant is from nest n_i. Let the number of pairs of the first type be $Z_1(r)$ and the number of pairs of the second type be $Z_2(r)$ (both random variables). Note that each of the variables $Z_1(r)$ and $Z_2(r)$ can be expressed as a sum of negatively-correlated 0/1 random variables, where each variable in each sum corresponds to one recruitment pair. For example, $Z_1(r)$ is a sum of negatively-correlated 0/1 random variables because the more likely it is for some ant from nest n_i to be the first ant in a recruitment pair, the less likely this is for another ant from nest n_i. Therefore, we can bound each of $Z_1(r)$ and $Z_2(r)$ by a Chernoff bound (Theorem 4.3 in [46]).

The number of recruitment pairs of the first type is equal to the total number of successful recruitment pairs involving ants from nest n_i minus the recruitment pairs involving two ants from nest n_i. So, by linearity of expectation, we have:

$$\begin{aligned}
\mathbb{E}\left[Z_1(r)\right] &\geq c(i,r-1)p(i,r-1)\alpha(r) - c(i,r-1)p(i,r-1)^2\alpha(r) \\
&= c(i,r-1)p(i,r-1)\alpha(r)(1-p(i,r-1)) \\
&\geq \frac{np(i,r-1)^2}{16}\left(1-1+p(j,r-1)\right) && \text{since } p(i,r-1)+p(j,r-1) \leq 1 \\
&\geq \frac{n\Sigma(r-1)^2}{16d^2}\left(\frac{\Sigma(r-1)}{d}\right) && \text{since } p(i,r-1), p(j,r-1) \geq \frac{\Sigma(r-1)}{d} \\
&\geq \frac{n}{16d^3k^3} && \text{since } \Sigma(r-1) \geq \frac{1}{k}.
\end{aligned}$$

Similarly (invoking the same results as above), the expected number of recruitment pairs of the second type is:

$$\mathbb{E}\left[Z_2(r)\right] \geq c(i,r-1)\Sigma(r-1)\alpha(r) - c(i,r-1)p(i,r-1)^2\alpha(r) \geq \frac{n}{16d^3k^3}.$$

Recall that $m = 128\sqrt{(c+7)\log n/n}$. By a Chernoff bound:

$$\Pr\left[|Z_1(r) - \mathbb{E}\left[Z_1(r)\right]| > \frac{mn}{256k}\right] \leq \frac{1}{n^{c+7}},$$
$$\Pr\left[|Z_2(r) - \mathbb{E}\left[Z_2(r)\right]| > \frac{mn}{256k}\right] \leq \frac{1}{n^{c+7}}.$$

By the definitions of $Z_1(r)$ and $Z_2(r)$, we have $c(i,r) = c(i,r-1) + Z_1(r) - Z_2(r)$. By linearity of expectation and the triangle inequality:

$$\begin{aligned}
&|c(i,r) - \mathbb{E}\left[c(i,r)\right]| \\
&\quad = |c(i,r-1) + Z_1(r) - Z_2(r) - c(i,r-1) - \mathbb{E}\left[Z_1(r)\right] + \mathbb{E}\left[Z_2(r)\right]| \\
&\quad \leq |Z_1(r) - \mathbb{E}\left[Z_1(r)\right]| + |Z_2(r) - \mathbb{E}\left[Z_2(r)\right]|.
\end{aligned}$$

Union-bounding over the events $|Z_1(r) - \mathbb{E}\left[Z_1(r)\right]| > mn/(256k)$ and $|Z_2(r) - \mathbb{E}\left[Z_2(r)\right]| > mn/(256k)$, we get that $\Pr\left[|c(i,r) - \mathbb{E}\left[c(i,r)\right]| < mn/(128k)\right] \geq 1 - 1/n^{c+6}$. Since $c(i,r) = np(i,r)$, the lemma holds. □

In the next lemma, we show that, assuming $Y(i,j,r-1)$ is small (that is, the gap between the nests is less than m and both nests are fairly large), the probability that $Y(i,j,r)$ is greater than $5m$ is very small (less than $1/n^2$). In other words, it is extremely unlikely that $Y(i,j,r)$ grows a lot in a single round.

Lemma 3.4.11. *For each pair of nests n_i and n_j such that $Y(i,j,r-1) < m$, $\Pr\left[Y(i,j,r) \geq 5m\right] \leq 1/n^2$.*

Proof. Fix an arbitrary pair of nests n_i and n_j such that $Y(i,j,r-1) < m$. By Claim 3.4.7, $p(i,r-1), p(j,r-1) \geq \Sigma(r-1)/d$ and $|p(i,r-1) - p(j,r-1)| < m$. Assume without loss of generality that $p(i,r-1) \geq p(j,r-1)$.

By Lemma 3.4.5 applied to both nests n_i and n_j, we have:

$$\begin{aligned}
&|\mathbb{E}\left[p(i,r) - p(j,r)\right]| \\
&\quad = |(p(i,r-1) - p(j,r-1))(1 + \alpha(r)(p(i,r-1) + p(j,r-1) - \Sigma(r-1)))| \\
&\quad \leq |2(p(i,r-1) - p(j,r-1))| \quad \text{since } p(i,r-1) + p(j,r-1) \leq 1, \alpha(r) \leq 1 \\
&\quad \leq 2m \qquad\qquad\qquad\qquad\quad \text{since } 0 \leq p(i,r-1) - p(j,r-1) \leq m.
\end{aligned}$$

By Lemma 3.4.10 and by a union bound, with probability at least $1 - 1/n^2$, it is true that $|p(i,r) - \mathbb{E}\left[p(i,r)\right]| \leq m$ and $|p(j,r) - \mathbb{E}\left[p(j,r)\right]| \leq m$.

By the triangle inequality, we have:

$$\begin{aligned}
&|p(i,r) - p(j,r)| \\
&\qquad = |p(i,r) - \mathbb{E}\left[p(i,r)\right] - p(j,r) + \mathbb{E}\left[p(j,r)\right] + \mathbb{E}\left[p(i,r) - p(j,r)\right]| \\
&\qquad \leq |p(i,r) - \mathbb{E}\left[p(i,r)\right]| + |p(j,r) - \mathbb{E}\left[p(j,r)\right]| + |\mathbb{E}\left[p(i,r) - p(j,r)\right]|.
\end{aligned}$$

Combining the results above, we get that with probability at least $1 - 1/n^2$, $|p(i,r) - p(j,r)| \leq 4m$. As a result of round r, it is also possible that one of the nests drops below the $\Sigma(r)/d$ threshold, resulting in $I(i,j,r) = 1$. This can add at most m more to the value of $Y(i,j,r)$. Therefore, we have $\Pr\left[Y(i,j,r) \geq 5m\right] \leq 1/n^2$. □

For the following results, we consider the probability distribution over all possible executions of Algorithm 7.

Next, we use Lemmas 3.4.9 and 3.4.11 in order to prove two properties of the $Y(i,j,r)$ variables over an arbitrary probabilistic execution of Algorithm 7: (1) (positive drift) whenever the random variable is below the target value, it increases in expectation by at least some fixed amount, and (2) (bounded jumps) the first time the random variable exceeds the target value, it does not "overshoot" the target value by too much.

Lemma 3.4.12 (Positive drift)**.** *For each time $r-1 > 0$, each pair of nests n_i and n_j, and each fixed value $m' < m$, $\mathbb{E}\left[Y(i,j,r) \mid Y(i,j,r-1) = m'\right] \geq m' + 1/(16d'k^3\sqrt{n})$.*

Proof. Fix an arbitrary time $r-1 > 0$, a pair of nests n_i and n_j, and a value $m' < m$. Note that Lemma 3.4.9 holds for an arbitrary time $r-1$, such that $Y(i,j,r-1) < m$, regardless of the execution preceding time $r-1$. By Lemma 3.4.9 and the law of total probability, we have that $\mathbb{E}\left[Y(i,j,r) \mid Y(i,j,r-1) = m'\right] \geq m' + 1/(16d'k^3\sqrt{n})$. □

Let $\tau(i,j)$ be the first time r in which $Y(i,j,r) \geq m$ (or ∞, otherwise).

Lemma 3.4.13 (Bounded jumps)**.** *For each pair of nests n_i and n_j,* $\Pr\left[Y(i,j,\tau(i,j)) \geq 5m\right] \leq 1/n$.

Proof. Fix an arbitrary pair of nests n_i and n_j. Note that Lemma 3.4.11 holds for an arbitrary time $r-1$, such that $Y(i,j,r-1) < m$, regardless of the execution preceding time $r-1$. Next, we sum over all possible fixed values for $\tau(i,j)$ (from 1 to ∞):

$$
\begin{aligned}
&\Pr\left[Y(i,j,\tau(i,j)) \geq 5m\right] \\
&\quad= \sum_{r=1}^{\infty} \Pr\left[Y(i,j,r) \geq 5m \mid \tau(i,j) = r\right] \cdot \Pr\left[\tau(i,j) = r\right] \\
&\quad\leq \sum_{r=1}^{\infty} \Pr\left[Y(i,j,r) \geq 5m \mid \tau(i,j) = r\right] \\
&\quad= \sum_{r=1}^{\infty} (\Pr\left[Y(i,j,r) \geq 5m \mid Y(i,j,t) < m \text{ for all } t \in [1,r-1]\right] \\
&\qquad\qquad \cdot \Pr\left[Y(i,j,t) < m \text{ for all } t \in [1,r-1]\right]) \\
&\quad= \sum_{r=1}^{\infty} (\Pr\left[Y(i,j,r) \geq 5m \mid Y(i,j,r-1) < m\right] \\
&\qquad\qquad \cdot \Pr\left[Y(i,j,t) < m \text{ for all } t \in [1,r-1]\right]) \\
&\quad= \sum_{r=1}^{\infty} (\Pr\left[Y(i,j,r) \geq 5m \mid Y(i,j,r-1) < m\right] \\
&\qquad\qquad \cdot \prod_{t=1}^{r-1} \Pr\left[Y(i,j,t) < m \mid Y(i,j,t-1) < m\right]).
\end{aligned}
$$

By Lemma 3.4.11, $\Pr\left[Y(i,j,r) \geq 5m \mid Y(i,j,r-1) < m\right] \leq 1/n^2$. Next, we upper bound $\Pr\left[Y(i,j,t) < m \mid Y(i,j,t-1) < m\right]$:

$$\begin{aligned}
&\Pr\left[Y(i,j,t) < m \mid Y(i,j,t-1) < m\right] \\
&\quad = 1 - \Pr\left[Y(i,j,t) \geq m \mid Y(i,j,t-1) < m\right] \\
&\quad \leq 1 - \Pr\left[p(i,t) < \frac{\Sigma(t-1)}{d} \middle| Y(i,j,t-1) < m\right].
\end{aligned}$$

Similarly to Lemma 3.4.10, we can express $c(i,t)$ as $c(i,t-1)+Z_1(t)-Z_2(t)$, where random variables $Z_1(t)$ and $Z_2(t)$ represent the number of new ants joining nest n_i in round t and the number of ants recruited away from nest n_i in round t, respectively. In Lemma 3.4.10, we applied a Chernoff bound to each of these variables because each one can be expressed as the sum of negatively-correlated binary random variables. Note that under certain conditions, the Chernoff bound is tight up to constants (see Theorem A.1.7 (reverse Chernoff bound) in the Appendix), so here, we will apply a reverse Chernoff bound to each of $Z_1(t)$ and $Z_2(t)$.

So far, we have:

$$\begin{aligned}
\mathbb{E}\left[Z_1(t) \mid Y(i,j,t-1) < m\right] &\geq \frac{n}{16d^3k^3}, \\
\mathbb{E}\left[Z_2(t) \mid Y(i,j,t-1) < m\right] &\geq \frac{n}{16d^3k^3}.
\end{aligned}$$

By Theorem A.1.7 and the assumption that $k \leq 64((c+7)n/\log n)^{1/4}$, we get:

$$\begin{aligned}
\Pr\left[Z_1(t) < 1 \mid Y(i,j,t-1) < m\right] &\geq \frac{1}{\sqrt{n}}, \\
\Pr\left[Z_2(t) > n(p(i,t-1) - \frac{\Sigma(t-1)}{d}) \mid Y(i,j,t-1) < m\right] &\geq \frac{1}{\sqrt{n}}.
\end{aligned}$$

By a union bound we have that, conditioning on $Y(i,j,t-1) < m$, with probability at least $1/n$, both $Z_1(t) < 1$ and $Z_2(t) > n(p(i,r-1)-\Sigma(r-1)/d)$, indicating that with probability at least $1/n$, it is true that $c(i,t) = c(i,t-1)+Z_1(t)-Z_2(t) \leq n\Sigma(t-1)/d$.

Therefore, since $p(i,t) = c(i,t)/n$, we have:

$$\begin{aligned}
&\Pr\left[Y(i,j,t) < m \mid Y(i,j,t-1) < m\right] \\
&\quad \leq 1 - \Pr\left[p(i,t) < \frac{\Sigma(t-1)}{d} \middle| Y(i,j,t-1) < m\right] \\
&\quad \leq 1 - \frac{1}{n}.
\end{aligned}$$

Finally, we get:

$$\Pr\left[Y(i,j,\tau(i,j)) \geq 5m\right] \leq \sum_{r=1}^{\infty} \left(\frac{1}{n^2}\right) \prod_{t=1}^{r-1} \left(1 - \frac{1}{n}\right) \leq \frac{1}{n}.$$

□

In order to bound the expected value of $\tau(i,j)$, we use a lemma from [13], which in turn uses Doob's Optional Stopping Theorem ([38]). Here, we restate the lemma from [13]. Intuitively, the following lemma analyzes the expected time until some arbitrary random variable (derived as a function of the state of a Markov chain) exceeds a fixed target value. The lemma requires the random variable to satisfy the positive drift and bounded jumps properties.

Lemma 3.4.14 (Lemma 3.2 in [13])**.** *Let $\{X_r\}_r$ be a Markov chain with finite state space Ω, and let $f : \Omega \to [0,n]$ be a function mapping states of the chain to non-negative integers. Let $\{Y_r\}_r$ be the stochastic process over N defined by $Y_r = f(X_r)$. Let $m \in N$ be a "target value", and let:*

$$\tau = \inf\{r \in \mathbb{N} : Y_r \geq m\}$$

be the random variable denoting the first time Y_r reaches or exceeds the value m. Assume that, for every state $x \in \Omega$ with $f(x) \leq m-1$, it holds that:

1. *(Positive drift) $\mathbb{E}\left[Y_{r+1} \mid X_r = x\right] \geq f(x) + \lambda$ for some $\lambda > 0$.*

2. *(Bounded jumps) $\Pr\left[Y_\tau \geq \alpha m \mid X_0 = x\right] \leq \frac{\alpha m}{n}$ for some $\alpha > 1$.*

Then, for every state $x \in \Omega$, it holds that:

$$\mathbb{E}\left[\tau \mid X_0 = x\right] \leq \frac{2\alpha m}{\lambda}.$$

Next, we would like to apply this lemma to our random variable $Y(i,j,r)$ in order to bound the value of $\tau(i,j)$. To establish a parallel between Lemma 3.4.14 and our setting, each state X_r of the Markov chain in the lemma corresponds to a mapping from the set of ants to the set of integers $\{0, \cdots, k\}$; this mapping represents the location (nest) of each ant at time r. In our setting, the function f translates the algorithm state to the value of the variable $Y(i,j,r)$ for a each pair of nests n_i and n_j (that is, Y_r corresponds to $Y(i,j,r)$ for nests n_i and n_j).

We already showed that Lemmas 3.4.12 and 3.4.13 yield the positive drift and bounded jumps properties for random variable $Y(i,j,r)$, as required by Lemma 3.4.14. Next, we apply Lemma 3.4.14 to random variable $Y(i,j,r)$, and use a Markov bound to show that, with high probability, the population gap between each pair of nests exceeds the m threshold within $\mathcal{O}(k^3 \log^{1.5} n)$ rounds.

Lemma 3.4.15. *Let $r_1 = 2^{14} d'(c+7)(c+3)k^3 \log^{1.5} n$. For each pair of nests n_i and n_j, $\Pr\left[Y(i,j,r_1) \geq m\right] \geq 1 - 1/n^{c+3}$.*

Proof. Fix an arbitrary pair of nests n_i and n_j. By Lemmas 3.4.12 and 3.4.13, we can apply Lemma 3.4.14 in order to bound the value of $\tau(i,j)$. In particular, we can use parameters $m = 128\sqrt{(c+7)\log n/n}$, $\lambda = 1/(16 d' k^3 \sqrt{n})$, and $\alpha = 5$. Therefore,

$$\mathbb{E}\left[\tau(i,j)\right] \leq \frac{\alpha m}{\lambda} \leq 2^{13} d'(c+7) k^3 \sqrt{\log n}.$$

By a Markov bound, $\Pr\left[\tau(i,j) \geq 2^{14} d'(c+7)k^3\sqrt{\log n}\right] \leq 1/2$. Using $(c+3)\log n$ repeated independent trials, where each trial is a sequence of $2^{14} d'(c+7)k^3\sqrt{\log n}$ algorithm rounds[4], we can boost this probability to $\Pr\left[\tau(i,j) \geq r_1\right] \leq 1/n^{c+3}$, indicating that $\Pr\left[Y(i,j,r_1) \geq m\right] \geq 1 - 1/n^{c+3}$. □

[4]The trials are independent because at the beginning of each trial we do not assume anything about the value of $Y(i,j,r)$ at the end of the previous trial; it can be arbitrarily small.

Recall that $d' = 2^8$.

Corollary 3.4.16. *Let* $r_1 = 2^{14}d'(c+7)(c+3)k^3 \log^{1.5} n$. *For all pairs of nests* n_i *and* n_j, $\Pr[Y(i,j,r_1) \geq m] \geq 1 - 1/n^{c+2}$.

Proof. Follows by a union bound from Lemma 3.4.15 since there are at most $\binom{k}{2} < k^2 < n$ pairs of nests. □

Corollary 3.4.16 is the key result of Section 3.4.4, and we will use it in Theorem 3.4.1 to analyse the final runtime of Algorithm 7. We will also use the helper Lemma 3.4.10 in the next section to reason about the concentration properties of the populations of nests.

3.4.5 Drop-out Stage

In this section, we focus on the gap between each pair of nests once the gap has reached the $m = \Omega(\sqrt{\log n/n})$ threshold. First, in Lemmas 3.4.17 and 3.4.18 we show that, once a nest drops below the $\Sigma(r)/d$ threshold, it loses all its ants within $\mathcal{O}(k \log n)$ rounds. Then, we show in Lemma 3.4.19 that, once the gap between two nests is at least m, the gap grows with high probability until one of the nests drops below the threshold of $\Sigma(r)/d$. In Corollary 3.4.20, we use a union bound in order to extend the result of Lemma 3.4.19 to all pairs of nests. Finally, in Theorem 3.4.1, we show that with high probability, after $\mathcal{O}(k \log n)$ rounds there is at most one surviving nest, indicating that the house hunting problem is solved in $\mathcal{O}(k^3 \log^{1.5} n)$ rounds (to grow the gap between all pairs of nests) plus $\mathcal{O}(k \log n)$ rounds (until all but one nest drop out).

For the results in this section, except the proof of Theorem 3.4.1, consider a fixed execution of Algorithm 7 and an arbitrary fixed time $r - 1 > 0$ in that execution. We consider the state variables at time $r - 1$ to be fixed, and we consider the probability distribution over the randomness in the next r_2+1 rounds, where $r_2 = 64(c+6)k \log n$.

First, we show that once a nest becomes small (with population proportion less than $\Sigma(r-1)/d$), with high probability it remains small in the subsequent rounds.

Lemma 3.4.17. *For each nest n_i with $p(i, r-1) < \Sigma(r-1)/d$, with probability at least $1 - 1/n^{c+5}$, $p(i, r') < \Sigma(r-1)/d$ for all times $r' \in [r, r + r_2]$.*

Proof. Fix an arbitrary nest n_i with $p(i, r-1) < \Sigma(r-1)/d$. First, we show that $p(i, r) < \Sigma(r-1)/d$ with probability at least $1 - 1/n^{c+6}$. Consider two possible cases:

Case 1: $p(i, r-1) < \Sigma(r-1)/(2d)$. Then, even if all ants successfully recruit, the nest cannot more than double in size, so $p(i, r) < \Sigma(r-1)/d$.

Case 2: $p(i, r-1) \geq \Sigma(r-1)/(2d)$. The expected number of ants from nest n_i that attempt to recruit in round r is:

$$c(i, r-1)p(i, r-1) \geq \frac{n(\Sigma(r-1))^2}{4d^2} \qquad \text{since } p(i, r-1) \geq \Sigma(r-1)/(2d).$$

The recruitment bits are set independently for each ant, so we can apply a Chernoff bound to bound the number of attempted recruitments by the ants in nest n_i. By the assumption that $k \leq 64((c+7)n/\log n)^{1/4}$ and by a Chernoff bound, we have that with probability at least $1 - 1/n^{c+7}$, at most $n(\Sigma(r-1))^2/(4.5d^2)$ ants attempt to recruit in round r.

The expected the number of ants to be recruited away from nest n_i is at most:

$$\begin{aligned}
&c(i, r-1)\left(1 - p(i, r-1)\right)\Sigma(r-1)\alpha(r) \\
&\geq \frac{n\Sigma(r-1)}{2d}\left(1 - \frac{\Sigma(r-1)}{d}\right)\Sigma(r-1) \quad \text{since } \frac{\Sigma(r-1)}{d} > p(i, r-1) \geq \frac{\Sigma(r-1)}{2d} \\
&\geq \frac{n(\Sigma(r-1))^2}{2d} - \frac{n(\Sigma(r-1))^3}{2d^2} \\
&\geq \frac{n(\Sigma(r-1))^2}{8d}.
\end{aligned}$$

Ants from nest n_i do not get recruited away from nest n_i independently, but they are negatively correlated (the more likely it is for some ant to get recruited away from nest n_i, the less likely it is for another ant to get recruited away form nest n_i). Since $k \leq 64((c+7)n/\log n)^{1/4}$, and by a Chernoff bound, with probability at least $1 - 1/n^{c+7}$, at least $n(\Sigma(r-1))^2/(2.5d^2)$ ants are recruited away from nest n_i.

By a union bound, with probability at least $1 - 1/n^{c+6}$, the number of ants at-

tempting to recruit is less than the number of ants recruited away from nest n_i, so the total population of n_i decreases, indicating $p(i, r) < \Sigma(r-1)/d$.

By a union bound over all $r_2 + 1 < n$ rounds, with probability at least $1 - 1/n^{c+5}$, $p(i, r') < \Sigma(r-1)/d$ for all times $r' \in [r, r + r_2]$. □

In the next lemma, we quantify how fast the population of a nest decreases once it drops below the $\Sigma(r-1)/d$ threshold.

Lemma 3.4.18. *For nest n_i with $p(i, r-1) < \Sigma(r-1)/d$, with probability at least $1 - 1/n^{c+4}$, $c(i, r + r_2) = 0$.*

Proof. Fix an arbitrary nest n_i. We calculate the expected change in the number of ants in nest n_i in round r by first calculating $\mathbb{E}[X_r^a]$ for some ant a in nest n_i. By Lemma 3.4.4:

$$\begin{aligned}
\mathbb{E}[X_r^a] &= \alpha(r)(p(i, r-1) - \Sigma(r-1)) \\
&\leq \left(\frac{1}{16}\right)\left(\frac{\Sigma(r-1)}{d} - \Sigma(r-1)\right) && \text{since } p(i, r-1) < \Sigma(r-1)/d \\
&\leq -\frac{1}{64k} && \text{since } d = \frac{4}{3}.
\end{aligned}$$

Therefore, by linearity of expectation and since the X_r^a variables of the ants in nest n_i are distributed identically:

$$\begin{aligned}
\mathbb{E}[p(i, r)] &= p(i, r-1) + \frac{1}{n} \sum_{\{a | \ell(a, r-1) = i\}} \mathbb{E}[X_r^a] \\
&\leq p(i, r-1)(1 + \mathbb{E}[X_r^a]) \\
&\leq p(i, r-1)\left(1 - \frac{1}{64k}\right).
\end{aligned}$$

By Lemma 3.4.17, $p(i, r') < \Sigma(r-1)/d$ for all times in $r' \in [r, r + r_2]$ with probability at least $1 - 1/n^{c+5}$. Therefore, it is true that $\mathbb{E}[c(i, r + r_2)] < 1/n^{c+5}$. By a Markov bound, nest n_i has at least one ant with probability at most $1/n^{c+5}$. Union-bounding over the events (1) $p(i, r') < \Sigma(r-1)/d$ for all times in $r' \in [r, r + r_2]$,

and (2) conditioning on (1), nest n_i has no ants at time $r + r_2$, we have that, with probability at least $1 - 1/n^{c+4}$, $c(i, r + r_2) = 0$. □

Next, we show that once the gap between two nests is fairly large (larger than m), at least one of the nests contains no ants within $\mathcal{O}(k \log n)$ rounds.

Lemma 3.4.19. *For each pair of nests n_i and n_j such that $Y(i, j, r-1) \geq m$, it is true that either $p(i, r + r_2) = 0$ or $p(j, r + r_2) = 0$ (or both) with probability at least $1 - 1/n^{c+3}$.*

Proof. Fix an arbitrary pair of nests n_i and n_j. We consider two cases based on the $I(i, j, r-1)$ component of $Y(i, j, r-1)$.

Case 1: $I(i, j, r-1) = 1$. Without loss of generality assume $p(i, r-1) < \Sigma(r-1)/d$. By Lemma 3.4.18, nest n_i drops out within r_2 rounds with probability at least $1 - 1/n^{c+4}$.

Case 2: $I(i, j, r-1) = 0$. Therefore, it must be true that $|p(i, r-1) - p(j, r-1)| \geq m$. Assume without loss of generality that $p(i, r-1) \geq p(j, r-1)$.

By Lemma 3.4.5 applied to both n_i and n_j, and since $p(i, r), p(j, r) \geq \Sigma(r-1)/d$:

$$\begin{aligned}\mathbb{E}\left[p(i, r) - p(j, r)\right] &\geq (p(i, r-1) - p(j, r-1)) \left(1 + \frac{\Sigma(r-1)}{32}\right) \\ &\geq (p(i, r-1) - p(j, r-1)) \left(1 + \frac{1}{32k}\right) \qquad \text{since } \Sigma(r-1) \geq 1/k.\end{aligned}$$

By Lemma 3.4.10:

$$\begin{aligned}\Pr\left[\left|p(i, r) - \mathbb{E}\left[p(i, r)\right]\right| \leq \frac{m}{128k}\right] &\geq 1 - \frac{1}{n^{c+6}}, \\ \Pr\left[\left|p(j, r) - \mathbb{E}\left[p(j, r)\right]\right| \leq \frac{m}{128k}\right] &\geq 1 - \frac{1}{n^{c+6}}.\end{aligned}$$

Therefore, with probability at least $1 - 1/n^{c+5}$ we have that $|p(i, r) - \mathbb{E}\left[p(i, r)\right]| \leq m/(128k)$ and $|p(j, r) - \mathbb{E}\left[p(j, r)\right]| \leq m/(128k)$.

By the reverse triangle inequality:

$$|p(i, r) - p(j, r)| \geq |\mathbb{E}\left[p(i, r) - p(j, r)\right]| - |\mathbb{E}\left[p(i, r)\right] - p(i, r)| - |\mathbb{E}\left[p(j, r)\right] - p(j, r)|.$$

So, with probability at least $1 - 1/n^{c+5}$, we have:

$$\begin{aligned}
&|p(i,r) - p(j,r)| \\
&\quad \geq (p(i,r-1) - p(j,r-1)) \left(1 + \frac{1}{32k}\right) - \frac{2m}{128k} \\
&\quad > (p(i,r-1) - p(j,r-1)) \left(1 + \frac{1}{64k}\right) \qquad \text{since } p(i,r-1) - p(j,r-1) \geq m.
\end{aligned}$$

So, we have that with probability at least $1 - 1/n^{c+5}$, the value of $|p(i,r) - p(j,r)|$ increases by at least a $(1 + 1/(64k))$ factor in one round. Union-bounding over all $r_2 + 1 < n$ rounds, we get that with probability at least $1 - 1/n^{c+4}$, $p(i, r + r_2) = 0$ or $p(j, r + r_2) = 0$. □

Next, we union-bound over all pairs of nests to show that, with high probability, in $r_2 + 1$ rounds there is exactly one nest with a non-zero population.

Corollary 3.4.20. *Suppose that for all pairs of nests n_i and n_j, it is true that $Y(i, j, r-1) > m$. Then, with probability at least $1 - 1/n^{c+2}$, there exists exactly one nest n_x such that $p(x, r + r_2) > 0$.*

Proof. By Lemma 3.4.19, for each pair of nests n_i and n_j with $Y(i, j, r-1) > m$, with probability at least $1 - 1/n^{c+3}$, at least one of the nests is empty after $r_2 + 1$ rounds. We know that $Y(i, j, r-1) > m$ for all pairs of nests n_i and n_j. Since there are at most $\binom{k}{2} < k^2 < n$ pairs of nests, by a union bound, with probability at least $1 - 1/n^{c+2}$, for all pairs of nests n_i and n_j, it is true that either $p(i, r + r_2) = 0$ or $p(j, r + r_2) = 0$. Therefore, with probability at least $1 - 1/n^{c+2}$, $p(x, r + r_2) > 0$ for at most one nest n_x. Since it is not possible for all nests to have a population proportion of 0, it must be the case, that, with probability at least $1 - 1/n^{c+2}$, there exists one nest n_x such that $p(x, r + r_2) \neq 0$, and all other nests have a population of 0. □

Finally, we are ready to resume the proof of Theorem 3.4.1. We assume an arbitrary probabilistic execution of Algorithm 7. Recall that $r_1 = 2^{14} d'(c+7)(c+3)k^3 \log^{1.5} n$ and $r_2 = 64(c+6)k \log n$.

Proof of Theorem 3.4.1. By the assumption that $k \leq 64((c+7)n/\log n)^{1/4}$, with probability at least $1 - 1/n^{c+1}$, in the first round of the algorithm (a round of searching), at least one ant finds a nest with quality 1. Since only ants at nests with quality 1 choose to recruit, at any point, at least one ant is recruiting to a good nest.

Next, we consider the events: (1) $Y(i, j, r_1) > m$ for all pairs of nests n_i and n_j, and (2) conditioning on (1), there exists exactly one nest n_x such that $p(x, r_1+r_2) > 0$. By Corollaries 3.4.16 and 3.4.20, and by the law of total probability, we get that with probability at least $1 - 1/n^{c+1}$, there is exactly one nest with a non-zero population by time $r_1 + r_2 \leq 2^{22}(c+7)(c+3)k^3 \log^{1.5} n = \mathcal{O}(k^3 \log^{1.5} n)$.

Finally, we consider the events: (1) at least one good nest is found in the first round of searching, and (2) conditioning on (1), there is exactly one nest with non-zero population by time $r_1 + r_2$. By the law of total probability, we get that with probability at least $1-1/n^c$, there is exactly one nest with non-zero population by time $r_1 + r_2$. Therefore, the house hunting problem is solved in $\mathcal{O}(k^3 \log^{1.5} n)$ rounds. □

3.5 House Hunting under Uncertainty

In the previous section we showed that Algorithm 7 solves the house-hunting problem in $\mathcal{O}(k^3 \log^{1.5} n)$ rounds. This result is somewhat far from the optimal running time of $\mathcal{O}(\log n)$, as demonstrated by our lower bound in Section 3.2 and optimal algorithm in Section 3.3. The reason why we introduced Algorithm 7 in the first place was to show that house hunting is solvable by a very simple and natural algorithm that is not susceptible to small perturbations of the environment parameters. In particular, the environment parameter we consider is the number of ants at each candidate nest, and the perturbations are small deviations of these populations from the exact values.

The rest of this section is organized as follows. In Section 3.5.1, we consider a model of uncertainty where a strong adversary determines the population estimates of each ant arbitrarily from some range of values. We give counterexamples to show that Algorithm 6 is not correct and Algorithm 7 is not efficient in this model. Then, in Section 3.5.2, we define a weaker adversary that chooses a family of distributions

from which the population estimates are drawn. We show that Algorithm 7 is correct in this model and we analyze its running time. Finally, in Section 3.5.3, we describe a density estimation algorithm [83] to provide ants with population estimates, and show how to compose this algorithm with Algorithm 7.

3.5.1 Algorithms 6 and 7 under a Strong Adversary

Strong adversarial model. Consider the following modification of the model in Section 3.1: the functions **search**(), **go**($\cdot$), and **recruit**($\cdot, \cdot$) return a pair $(j, \tilde{c_j})$ where j is a nest id, c_j is the number of ants at nest n_j, and $\tilde{c_j}$ is an arbitrary integer in the range $\tilde{c_j} \in [(1-\epsilon)c_j, (1+\epsilon)c_j]$ for some ϵ, $0 < \epsilon < 1$. That is, instead of returning the actual population of the nest, the functions return an arbitrary value in some range around the actual population.

Example 3.5.1 (Counterexample for Algorithm 6 with uncertainty)**.** *Consider an arbitrary $\epsilon > 1/2$ and an arbitrary nest n_i. In the first round of the execution of Algorithm 6 (a round of searching), by a Chernoff bound, with high probability (at least $1 - 1/n^c$ for some constant $c > 1$), $1 < c(i, 1) < n$; that is, the population of nest n_i is more than one ant but not all ants. Since a nest cannot more than double in size in the second round (or in any other round), we know that in each execution $\lceil c(i, 2)/2 \rceil \leq c(i, 1)$. In the second round, the adversary is allowed to return any values in the range $[c(i, 2)(1 - \epsilon), c(i, 2)(1 + \epsilon)]$ as the population estimates of ants in nest n_i (potentially different estimates for different ants). The smallest integer in this range is $\lceil c(i, 2)(1-\epsilon) \rceil < \lceil c(i, 2)/2 \rceil \leq c(i, 1)$. In the second round, the adversary returns $\lceil c(i, 2)(1 - \epsilon) \rceil$ to each ant in nest n_i. Then, all ants nest n_i see a decrease in population compared to $c(i, 1)$ and the nest drops out of the competition. The adversary can do this for each nest and thus cause all nests to drop out in the second round of the execution. In this counterexample, we have shown that for any $\epsilon > 1/2$, with high probability, the adversary can cause all nests to drop out in the second round of the execution. Thus, with high probability, at any time in the execution, there are at least two nests that contain some ants. This contradicts the problem statement,*

which states that, with high probability, starting at some time in the execution there exists a unique nest that contains all ants.

Example 3.5.2 (Counterexample for Algorithm 7 with uncertainty)**.** *Consider an arbitrary $\epsilon \geq 1/2$. For simplicity, consider the case where there are only two nests and the number of ants is even. In the first round, by a Chernoff bound, with high probability (at least $1 - 1/n^c$ for some constant $c > 1$) the resulting populations $c(1,1)$ and $c(2,1)$ are both within a small constant factor of $n/2$. So, with high probability, the ranges $[(1-\epsilon)c(1,1), (1+\epsilon)c(1,1)]$ and $[(1-\epsilon)c(2,1), (1+\epsilon)c(2,1)]$ both include $n/2$. In the next round, suppose the adversary returns $n/2$ as the population estimate to all the ants. Since the two nests are close in population and all ants recruit with the same probability, again by a Chernoff bound, we can show that with high probability the nest populations still remain within a small constant factor of $n/2$. By a union bound, we can repeat this reasoning over polynomially many rounds in n to show that, with high probability, the populations of both nests remain within a constant factor of $n/2$, so the adversary can cause all ants to recruit with probability $1/2$ in all polynomially many rounds. Therefore, after polynomially many rounds, there is still not a single winning nest, indicating that for an arbitrary $\epsilon \geq 1/2$, with high probability, the running time of Algorithm 7 is $\mathcal{O}(poly(n))$ (an exponential increase compared to the $\mathcal{O}(\log^{1.5} n)$ running time for two nests and no uncertainty).*

Both examples above assume ϵ is fairly large (at least $1/2$). This leaves the possibility that Algorithms 6 and 7 work well if ϵ is very small (small amount of uncertainty). Consider an extreme case where $\epsilon < 1/n$; that is, the estimate $\tilde{c_j}$ of the population of nest n_j is within less than one ant away from the actual population c_j. Since $\tilde{c_j}$ is an integer, we are always guaranteed that $\tilde{c_j} = c_j$, and both algorithms behave identically to the model with no uncertainty. At this point, we have no counterexamples or analysis to identify the exact thresholds (between $1/n$ and $1/2$) of ϵ for which the algorithms tolerate uncertainty. However, in Theorem 3.5.3, we show that in the weak adversarial model, even for large ϵ (at least $1/2$), Algorithm 7 is correct and only a constant factor slower compared to the case of no uncertainty.

3.5.2 Algorithm 7 under a Weak Adversary

In this section, we restrict the power of the adversary in such a way that the population estimates are not chosen arbitrarily from a range but from a family of distributions that satisfy certain properties. Then, we analyze the correctness and running time of Algorithm 7 under this weak adversarial model.

3.5.2.1 Weak adversarial model

First, we define a family of probability distributions that we use in the definition of the weak adversarial model. An (ϵ, c', n)-family $\mathcal{F}$ of distributions, where $0 < \epsilon < 1$, $c' > 2$, $n \in \mathbb{N}$, is a set of distributions $\mathcal{F} = \{F_x\}_{x \in \mathbb{N}}$, where each F_x is an x-variate distribution. For each $a \in [1, x]$ let F_x^a denote the marginal probability distribution of the a'th element of F_x. Then, for each $a \in [1, x]$, the following are satisfied:

1. $\sum_{y \in [(1-\epsilon)x, (1+\epsilon)x]} F_x^a(y) \geq 1 - 1/n^{c'}$,
2. F_x^a has mean x,
3. the domain of F_x^a is $[0, n]$.

In other words, sampling from distribution F_x in the (ϵ, c', n)-family of distributions results in a vector of x values, each of which is: (1) in the range $[(1-\epsilon)x, (1+\epsilon)x]$ with probability at least $1-1/n^{c'}$, (2) equal to x in expectation, and (3) guaranteed to be in the range $[0, n]$. The different values in the vector are not assumed to be chosen independently. Note that (1) is similar to the definition of the strong adversary in Section 3.5.1; however, here the distributions from which the adversary chooses the population estimates are determined ahead of time, at the beginning of the execution, and depend only on the size of the nest.

Consider the following modification of the model in Section 3.1. For each nest n_j with population c_j, and for any constant $c' > 2$ and any ϵ, such that $0 < \epsilon < 1$, let F_{c_j} be a distribution from a (ϵ, c', n)-family of distributions, and let y be a vector of size c_j sampled from F_{c_j}. For each ant a in nest n_j, the functions **search**(), **go**($\cdot$), and **recruit**($\cdot, \cdot$) return a pair $(j, \tilde{c_j})$ where $\tilde{c_j}$ is the element of the vector y that corresponds to ant a.

In other words, with probability at least $1 - 1/n^{c'}$, the estimate $\tilde{c_j}$ of each ant is within a range (that depends on ϵ) around the exact value c_j, the estimate is correct in expectation, and it is upper bounded by the number n of ants. The adversary chooses the distribution from which the population estimates $\tilde{c_j}$ are drawn with knowledge of the actual population c_j of the nest. Therefore, for different nest populations, the adversary chooses different distributions. Also, we have not made any independence assumptions, so it is possible that the adversary chooses population estimates that are correlated between different ants. One example of such correlation is returning a larger population estimate to some ant provided that another ant in the same nest has a large population estimate. This type of correlation is what we observe in the density estimation algorithm that we will discuss in Section 3.5.3; the more two ants collide with each other, the more both of their estimates of the population increase.

3.5.2.2 Algorithm 7 in the weak adversarial model

Next, we analyze Algorithm 7 under the weak adversarial model defined above. We prove the following main result.

Theorem 3.5.3. *For any constants c and c', such that $2 < c' < c$, any ϵ, such that $0 < \epsilon < 1$, and any (ϵ, c', n)-family $\mathcal{F}$ of distributions, with probability at least $1 - 1/n^c - 1/n^{c'-2}$, Algorithm 7 using $\mathcal{F}$ solves the* HouseHunting *problem in the weak adversarial model in $(2^{22}(c+7)(c+3)k^3 \log^{1.5} n)/(1-\epsilon)^2 = \mathcal{O}(1/(1-\epsilon)^2)\mathcal{O}(k^3 \log^{1.5} n)$ rounds.*

Proof overview: The proof follows the same structure as the proof in the case of no uncertainty in Section 3.4.2. In particular, (almost) all results from Section 3.4.2 hold if we condition on the event that the population estimates of all ants for all rounds in the execution (of a fixed length) lie in the range of a $[(1-\epsilon), (1+\epsilon)]$ factor of the true populations. Then, in the proof of Theorem 3.5.3, we bound the probability of this event (by a union bound) to be at least $1 - 1/n^{c'-2}$. Finally, we combine this result (by the law of total probability) with the result about the correctness and running time of Algorithm 7 to get that with probability at least $1 - 1/n^c - 1/n^{c'-2}$ the HouseHunting problem is solved in the weak adversarial model.

Next, we fix the following parameters. Fix arbitrary constants c and c', such that $2 < c' < c$, and fix $d = 4/3$, and $d' = 2^8$. Also, fix an arbitrary ϵ, $0 < \epsilon < 1$ and an (ϵ, c', n)-family $\mathcal{F}$ of distributions.

The following assumptions and definitions are similar to the ones in Section 3.4.2. We assume $k \leq 64((c+7)n/((1-\epsilon)^2 \log n))^{1/4}$ (a factor of $1/(1-\epsilon)^2$ larger than the case of no uncertainty). Let random variable $c(i,r)$ be the population of nest n_i at time r, $p(i,r) = c(i,r)/n$ and $\Sigma(r) = \sum_{i=1}^{k} p(i,r)^2$. Note that in the proofs in Section 3.4.2, the value (or random variable) $p(i,r)$ is used to represent two different concepts: the population proportion of nest n_i at time r, and the recruitment probability of ants in nest n_i in round $r+1$. The weak adversarial model affects only the second usage of $p(i,r)$ in the proofs.

In the weak adversarial model, all the ants located in some nest n_i at some time r obtain estimates $e(\cdot, i, r+1)$ of the value of $p(i,r)$ in round $r+1$ drawn from the $F_{c(i,r)}$ distribution (and converted to population proportions by dividing by n). Based on the properties of the (ϵ, c', n)-family of distributions and the fact that $c(i,r) = np(i,r)$, we have that, for each ant a:

1. $\Pr\left[e(i,a,r+1) \in [(1-\epsilon)\cdot p(i,r), (1+\epsilon)\cdot p(i,r)]\right] \geq 1 - 1/n^{c'}$,
2. $\mathbb{E}\left[e(i,a,r+1)\right] = p(i,r)$,
3. $e(a,i,r+1) \leq 1$.

Note that we do not make independence assumptions, so it is possible that the adversary chooses population estimates that are correlated between different ants. In particular, we assume that in a given round r and a given nest n_i, there may be correlations between the population estimates of the ants in nest n_i, but no correlations between the population estimates of ants in different nests and across different rounds. The independence between estimates across rounds is crucial for our analysis, while the independence between estimates in different nests is just a simplifying assumption. This assumption is inspired by thinking of the population estimates as being determined by a distributed randomized algorithm that runs independently at each nest and in each round.

First, we prove a simple property that follows from the assumptions above that we will use in the subsequent proofs.

For each time r and each ant a such that $\ell(a, r) = i$, let $E(a, i, r + 1)$ denote the event that $e(a, i, r + 1) \in [(1 - \epsilon)p(i, r), (1 + \epsilon)p(i, r)]$.

Lemma 3.5.0. *For each time r, each nest n_i, and each ant a such that $\ell(a, r) = i$:*

$$p(i,r) - \frac{1}{n^{c'-1}} \leq \mathbb{E}\left[e(a,i,r+1) \mid E(a,i,r)\right] \leq p(i,r) + \frac{1}{n^{c'-1}}.$$

Proof. By the law of total expectation, we have:

$$\begin{aligned}
&\mathbb{E}\left[e(a,i,r+1) \mid E(a,i,r+1)\right] \\
&\quad = \frac{\mathbb{E}\left[e(a,i,r+1)\right] - \mathbb{E}\left[e(a,i,r+1) \mid \bar{E}(a,i,r+1)\right] \cdot \Pr\left[\bar{E}(a,i,r+1)\right]}{\Pr\left[E(a,i,r+1)\right]}.
\end{aligned}$$

We have that $\mathbb{E}\left[e(a, i, r + 1)\right] = p(i, r)$, $\mathbb{E}\left[e(a, i, r + 1) \mid \bar{E}(a, i, r + 1)\right] \leq 1$, and $Pr\left[E(a, i, r + 1)\right] \geq 1 - 1/n^{c'}$, so:

$$p(i,r) - \frac{1}{n^{c'-1}} \leq \mathbb{E}\left[e(a,i,r+1) \mid E(a,i,r+1)\right] \leq p(i,r) + \frac{1}{n^{c'-1}}.$$

□

Next, we restate the results from Section 3.4.2 and emphasize the differences in the proofs under the weak adversarial model. Most of the lemma and claim statements are similar or identical to the ones in Section 3.4.2. Here, we state the new results and refer to the corresponding results in Section 3.4.2. The lemma/theorem numbering in this section corresponds to the one in Section 3.4.2.

The main differences between the results in this section and the results in Section 3.4.2 can be classified as two types. First, in all statements and proofs in Section 3.4.2 that use only expected values and linearity of expectation (Lemmas 3.4.4, 3.4.5, 3.4.6, and 3.4.9), we use the bounds in Lemma 3.5.0 instead of the original values of the expectations of the $p(i, r)$ values. This introduces additive error terms of $1/n^{c'-2}$ that carry over throughout the statements and proofs until they get subsumed in

Lemma 3.5.9. Second, for stronger probabilistic results, we first condition on the event $E(a, i, r)$ for all ants a in nest n_i, then fix arbitrarily the population estimates chosen by the adversary, and use the lower bound $p(i, r)(1 - \epsilon)$ of these estimates in our calculations. Finally, we sum over all possible values of the population estimates using the law of total probability. The resulting $(1 - \epsilon)$ factors thus appear in the running time of the algorithm as a $1/(1 - \epsilon)^2$-factor increase.

For the following results, consider a fixed execution of Algorithm 7 (in the weak adversarial model) and an arbitrary fixed time $r - 1 > 0$ in that execution. We assume the state variables at time $r - 1$ to be fixed, and we consider the probability distribution over the randomness in round r. Moreover, assume that for each ant a, $e(a, i, r) \in [(1 - \epsilon)p(i, r - 1), (1 + \epsilon)p(i, r - 1)]$ (note that we are not fixing the value of $e(a, i, r)$, just its range).

Here we define random variable X_r^a the same way as in Section 3.4.2. That is, for each ant a and each time r, let random variable X_r^a take on values:

- 1 if a is the first element of a pair in the set of successful recruitments M, except if the pair is (a, a) (that is, a recruits itself),
- -1 if a is the second element of a pair in the set of successful recruitments M, except if the pair is (a, a),
- 0 if a does not appear in any recruitment pair in the set of successful recruitments M, or if $(a, a) \in M$.

Also, as in Section 3.4.2 we define $\alpha(a, r) = \Pr[X_r^a = 1 \mid b(a, r) = 1]$ and $\beta(a, r) = \Pr[X_r^a = -1]$ (where $b(a, r)$ is the recruitment bit of ant a in round r).

Lemma 3.5.2 (Corresponds to Lemma 3.4.2)**.** *For each pair of ants a and a', $\alpha(a, r) = \alpha(a', r) \geq 1/16$.*

Proof sketch. The probability $\alpha(a, r)$ is defined with respect to ants that are actively recruiting in round r, regardless of how they set their recruitment bits. Therefore, the same arguments from Lemma 3.4.2 hold here as well. □

Lemma 3.5.3 (Corresponds to Lemma 3.4.3)**.** *For each nest n_i:*

$$\beta(i,r) = \alpha(r)\sum_{i=1}^{k}\sum_{\{a|\ell(a,r)=i\}} p(i,r-1)e(a,i,r).$$

Proof sketch. The calculations in the proof are similar to the proof of Lemma 3.4.3. □

Lemma 3.5.4 (Corresponds to Lemma 3.4.4)**.** *For each nest n_i, we have:*

$$\alpha(r)\left(p(i,r-1) - \Sigma(r-1) - \frac{2}{n^{c'-1}}\right) \leq \mathbb{E}\left[X_r^a\right],$$
$$\alpha(r)\left(p(i,r-1) - \Sigma(r-1) + \frac{2}{n^{c'-1}}\right) \geq \mathbb{E}\left[X_r^a\right].$$

Proof. By definition, $\mathbb{E}\left[X_r^a\right] = \Pr\left[X_r^a = 1\right] - \Pr\left[X_r^a = -1\right]$. By Lemmas 3.4.2 and 3.5.3:

$$\mathbb{E}\left[X_r^a\right] = \alpha(r)e(a,i,r) - \alpha(r)\sum_{i=1}^{k}\sum_{\{a|\ell(a,r-1)=i\}} p(i,r-1)e(a,i,r).$$

By Lemma 3.5.0 and by linearity of expectation, it follows that

$$\begin{aligned}
\mathbb{E}\left[X_r^a\right] &\leq \alpha(r)\left(p(i,r-1) + \frac{1}{n^{c'-1}} - \sum_{i=1}^{k} p(i,r-1)\left(p(i,r-1) - \frac{1}{n^{c'-1}}\right)\right) \\
&= \alpha(r)\left(p(i,r-1) + \frac{1}{n^{c'-1}} - \Sigma(r-1) + \sum_{i=1}^{k} p(i,r-1)\left(\frac{1}{n^{c'-1}}\right)\right) \\
&= \alpha(r)\left(p(i,r-1) - \Sigma(r-1) + \frac{2}{n^{c'-1}}\right).
\end{aligned}$$

Similarly, we can show that $\mathbb{E}\left[X_r^a\right] \geq \alpha(r)(p(i,r-1) - \Sigma(r-1) - 2/n^{c'-1})$. □

Lemma 3.5.5 (Corresponds to Lemma 3.4.5)**.** *For each nest n_i, we have:*

$$\mathbb{E}\left[p(i,r)\right] \leq p(i,r-1)(1 + \alpha(r)\left(p(i,r-1) - \Sigma(r-1) + \frac{2}{n^{c'-1}}\right)$$
$$\mathbb{E}\left[p(i,r)\right] \geq p(i,r-1)(1 + \alpha(r)\left(p(i,r-1) - \Sigma(r-1) - \frac{2}{n^{c'-1}}\right)$$

Proof Sketch. The proof of Lemma 3.4.5 uses only linearity of expectation and Lemma 3.4.4. Therefore, by following the same calculations and applying Lemma 3.5.4, the lemma holds. □

Lemma 3.5.6 (Corresponds to Lemma 3.4.6)**.** *Let n_i and n_j be a pair of nests such that $p(i, r-1), p(j, r-1) \geq \Sigma(r-1)/d$. Then:*

$$\mathbb{E}\left[|p(i,r) - p(j,r)|\right] \geq |p(i,r-1) - p(j,r-1)| \left(1 + \frac{\Sigma(r-1)}{32} - \frac{4}{n^{c'-1}}\right).$$

Proof Sketch. The proof of Lemma 3.4.6 uses only linearity of expectation and Lemma 3.4.5. Therefore, by following the same calculations and applying Lemma 3.5.5, the lemma holds. □

Let $m = 128\sqrt{(c+7)\log n/(n(1-\epsilon)^2)}$ (the value of m is a factor of $1/\sqrt{(1-\epsilon)^2}$ larger than in the case of no uncertainty). As in Section 3.4.2, we define random variables $I(i,j,r)$ and $Y(i,j,r)$ for each time r and each pair of nests n_i and n_j:

$$I(i,j,r) = \begin{cases} 1, & \text{if } \min\{p(i,r), p(j,r)\} < \Sigma(r)/d, \\ 0, & \text{otherwise.} \end{cases}$$

$$Y(i,j,r) = |p(i,r) - p(j,r)| + m \cdot I(i,j,r).$$

Claim 3.5.7 (Corresponds to Claim 3.4.7)**.** *For each pair of nests n_i and n_j, if $Y(i,j,r) < m$ then $I(i,j,r) = 0$, $p(i,r), p(j,r) \geq \Sigma(r)/d$, and $|p(i,r) - p(j,r)| < m$.*

In Section 3.4.2, we define the equivalent Methods 1 and 2 for the recruitment process. Here, we can use the same definitions, the only difference being that the distribution of random variable B (the recruitment bits of the ants) here is different. However, note that the only property of the distribution of B we used is the fact that B is independent from G, P and P', which is also true here in the weak adversarial model. Therefore, Claim 3.4.8 holds here as well:

Claim 3.5.8 (Corresponds to Claim 3.4.8)**.** *The set M of recruitment pairs resulting from the triple of random variables (G, B, P) generated by Method 1 is identically distributed to the set M' of recruitment pairs resulting from the triple of random variables (G, B, P') generated by Method 2.*

Lemma 3.5.9 (Corresponds to Lemma 3.4.9)**.** *For each pair of nests n_i and n_j such that $Y(i, j, r-1) < m$, it is true that:*

$$\mathbb{E}\left[Y(i,j,r)\right] \geq Y(i,j,r-1) + \frac{1}{32d'k^3\sqrt{n}}.$$

Proof Sketch. The same arguments as in Lemma 3.4.9 work here as well with small modifications and by carrying the error terms of the expectations.

Case 1: When we bound the value of $|K|$ (the number of key pairs where the first ant is from nest n_i, the second ant is from nest n_j and both ants are actively recruiting), we use the fact that ants from nests n_i and n_j recruit with probabilities $p(i, r-1)$ and $p(j, r-1)$. Here, we need to use the estimates of these values: $e(\cdot, i, r-1)$ and $e(\cdot, j, r-1)$, respectively. Fix arbitrarily the values of $e(\cdot, i, r-1)$ and $e(\cdot, j, r-1)$ for all ants in nests n_i and n_j. We know that for each ant in nest n_i, this fixed value is at least $(1-\epsilon)p(i, r-1)$, and for each ant in nest n_j, the value is at least $(1-\epsilon)p(j, r-1)$. Once we have fixed the population estimates, the recruitment bits of all ants from nests n_i and n_j are determined independently from each other, so we can use the same arguments as in the proof of Lemma 3.4.9 to conclude that the expected number of key pairs is at least $np(i, r-1)^2 p(j, r-1)^2 (1-\epsilon)^2 \alpha(r)$.

Note that we can still apply a Chernoff bound to bound the number of key pairs because determining the ants' recruitment bits is still done independently. Therefore, the number of key pairs can still be represented as a sum of binary pairwise negatively-correlated random variables and we can apply a Chernoff bound (Theorem 4.3 in [46]).

Since the above arguments hold for any fixed estimates of the populations, by the law of total probability, they also hold for the distribution of population estimates.

In order for the rest of the calculations to hold in this case, we need to use the new assumption that $k \leq 64((c + 7n/((1-\epsilon)^2 \log n)))^{1/4}$.

Case 2: Similarly to the proof of Lemma 3.4.9, we use Lemma 3.5.6 to bound the expectation of $|p(i,r) - p(j,r)|$. Recall that $c' > 2$ and $k \leq 64((c + 7n/((1 - \epsilon)^2 \log n)))^{1/4}$.

$$\begin{aligned}
\mathbb{E}\left[|p(i,r) - p(j,r)|\right] &\geq |p(i,r-1) - p(j,r-1)| \left(1 + \frac{\Sigma(r-1)}{32} - \frac{4}{n^{c'-1}}\right) \\
&\geq |p(i,r-1) - p(j,r-1)| + \left(\frac{2}{d'k^2\sqrt{n}}\right)\left(\frac{1}{32k} - \frac{4}{n^{c'-1}}\right) \\
&\geq |p(i,r-1) - p(j,r-1)| + \frac{1}{32d'k^3\sqrt{n}}.
\end{aligned}$$

□

Lemma 3.5.10 (Corresponds to Lemma 3.4.10)**.** *For each nest n_i such that $p(i, r-1) \geq \Sigma(r-1)/d$,* $\Pr\left[|p(i,r) - \mathbb{E}\left[p(i,r)\right]| \leq m/(128k)\right] \geq 1 - 1/n^{c+6}$.

Proof Sketch. The same arguments as in the proof of Lemma 3.4.10 work here with some small quantitative differences. When we bound the expected values of $Z_1(r)$ and $Z_2(r)$ (the number of new ants joining nest n_i and the number of ants recruited form nest n_i, respectively), we need to use the lower bound $(1 - \epsilon)p(i, r-1)$ of the recruitment probability $e(\cdot, i, r)$, similarly to the proof of Lemma 3.5.9. The calculations still hold due to the increased new value of m.

Similarly to the proof of Lemma 3.5.9, we can apply a Chernoff bound to the random variables $Z_1(r)$ and $Z_2(r)$ because they can still be expressed as sums of pairwise negatively correlated binary random variables. □

Lemma 3.5.11 (Corresponds to Lemma 3.4.11)**.** *For each pair of nests n_i and n_j such that $Y(i,j,r-1) < m$,* $\Pr\left[Y(i,j,r) \geq 5m\right] \leq 1/n^2$.

Proof Sketch. The proof of Lemma 3.4.11 holds here as well. □

For the following results, we consider the probability distribution over all possible executions of Algorithm 7 in the weak adversarial model. Since the following results apply to the entire probabilistic execution (as opposed to a distribution based on a single-round transition), we need the independence between the population estimates

of the ants across different rounds. In particular, we condition on the single-round results and then use the law of total probability to show they hold for the entire probabilistic execution.

Let $E(r)$ denote the event that for all nests n_i and all ants a such that $\ell(a, r) = i$, $e(a, i, r) \in [(1 - \epsilon)p(i, r - 1), (1 + \epsilon)p(i, r - 1)]$.

Lemma 3.5.12 (Corresponds to Lemma 3.4.12)**.** *For each time $r - 1 > 0$, each pair of nests n_i and n_j, and each fixed value $m' < m$, conditioning on event $E(r)$, $\mathbb{E}[Y(i, j, r) \mid Y(i, j, r - 1) = m'] \geq m' + 1/(32d'k^3\sqrt{n})$.*

Proof Sketch. The proof of Lemma 3.4.11 holds here as well, after adding the extra conditioning on event $E(r)$. □

Lemma 3.5.13 (Corresponds to Lemma 3.4.13)**.** *For each pair of nests n_i and n_j, conditioning on $E(r')$ for all rounds $r' \in [1, \tau(i, j)]$, $\Pr[Y(i, j, \tau(i, j)) \geq 5m] \leq 1/n$.*

Proof Sketch. The same arguments as in the proof of Lemma 3.4.13 work here with some small quantitative differences. When we bound the expected values of $Z_1(t)$ and $Z_2(t)$ (the number of new ants joining nest n_i and the number of ants recruited form nest n_i, respectively), we need to use the lower bound $(1 - \epsilon)p(i, r - 1)$ of the recruitment probability $e(\cdot, i, r)$, similarly to the proof of Lemma 3.5.9. The calculations still hold due to the increased new value of m. □

Lemma 3.5.15 (Corresponds to Lemma 3.4.15)**.** *For each pair of nests n_i and n_j, conditioning on event $E(r')$ for all rounds $r' \in [1, r_1]$, $\Pr[Y(i, j, r_1) \geq m] \geq 1 - 1/n^{c+3}$, where $r_1 = (2^{14}d'(c + 7)(c + 3)k^3 \log^{1.5} n)/(1 - \epsilon)^2$.*

Proof Sketch. The proof of Lemma 3.4.15 holds here as well, using the new increased values of m and r_1, and conditioning on event $E(r')$ for all rounds $r' \in [1, r_1]$. In order to apply Lemma 3.4.14 from [11], we need the independence of the population estimates of the ants across different rounds. □

Corollary 3.5.16 (Corresponds to Corollary 3.4.16)**.** *Let $r_1 = 2^{14}d'(c + 7)(c + 3)k^3 \log^{1.5} n$. For all pairs of nests n_i and n_j, conditioning on event $E(r')$ for all rounds $r' \in [1, r_1]$, $\Pr[Y(i, j, r_1) \geq m] \geq 1 - 1/n^{c+2}$.*

Let $r_2 = 64(c+6)k \log n/(1-\epsilon)^2$. For the following results, except the proof of Theorem 3.5.3, consider a fixed execution of Algorithm 7 and an arbitrary fixed time $r-1 > 0$ in that execution. We consider the state variables at time $r-1$ to be fixed, and we consider the probability distribution over the randomness in the next $r_2 + 1$ rounds.

Lemma 3.5.17 (Corresponds to Lemma 3.4.17)**.** *For each nest n_i with $p(i, r-1) < \Sigma(r-1)/d$, conditioning on event $E(r')$ for all rounds $r' \in [r, r+r_2]$, with probability at least $1 - 1/n^{c+5}$, $p(i, r') < \Sigma(r-1)/d$ for all times $r' \in [r, r+r_2]$.*

Proof Sketch. The same arguments as in the proof of Lemma 3.4.13 work here by using $(1-\epsilon)p(i, r-1)$ as a bound for the recruitment probability $e(\cdot, i, r)$, similarly to the proof of Lemma 3.5.9. The calculations still hold due to the new assumption of the relationship between n and k ($k \leq 64((c + 7n/((1-\epsilon)^2 \log n)))^{1/4}$). □

Lemma 3.5.18 (Corresponds to Lemma 3.4.18)**.** *For each nest n_i with $p(i, r-1) < \Sigma(r-1)/d$, conditioning on event $E(r')$ for all rounds $r' \in [r, r+r_2]$, with probability at least $1 - 1/n^{c+4}$, $c(i, r+r_2) = 0$.*

Proof Sketch. The proof of Lemma 3.4.18 holds here as well. □

Lemma 3.5.19 (Corresponds to Lemma 3.4.19)**.** *For each pair of nests n_i and n_j such that $Y(i, j, r-1) \geq m$, conditioning on event $E(r')$ for all rounds $r' \in [r, r+r_2]$, with probability at least $1 - 1/n^{c+3}$, either $p(i, r+r_2) = 0$ or $p(j, r+r_2) = 0$ (or both).*

Proof Sketch. The proof of Lemma 3.4.19 holds here as well. □

Corollary 3.5.20 (Corresponds to Corollary 3.4.20)**.** *Suppose that for all pairs of nests n_i and n_j, it is true that $Y(i, j, r-1) > m$. Then, conditioning on event $E(r')$ for all rounds $r' \in [r, r+r_2]$, with probability at least $1 - 1/n^{c+2}$, there exists exactly one nest n_x such that $p(x, r+r_2) > 0$.*

Finally, we are ready to resume the proof of Theorem 3.4.1. We assume an arbitrary probabilistic execution of Algorithm 7. Recall that $r_1 = (2^{14}d'(c+7)(c+3)k^3 \log^{1.5} n)/(1-\epsilon)^2$ and $r_2 = 64(c+6)k \log n/(1-\epsilon)^2$.

Proof of Theorem 3.5.3. Following the same arguments as in the proof of Theorem 3.4.1, we can conclude that, with probability at least $1 - 1/n^c$, conditioning on event $E(r')$ for all rounds $r' \in [1, r_1 + r_2]$, there is exactly one nest with non-zero population by time $r_1 + r_2$.

Consider the events: (1) event $E(r')$ for all rounds $r' \in [1, r_1 + r_2]$, and (2) conditioning on event $E(r')$ for all rounds $r' \in [1, r_1 + r_2]$, there is exactly one nest with non-zero population by time $r_1 + r_2$. By a union bound over all ants a and all rounds $r' \in [1, r_1 + r_2]$ (note that $r_1 + r_2 < n$), the probability of event (1) is at least $1 - 1/n^{c'-2}$. By the law of total probability (with respect to events (1) and (2)), with probability at least $1 - 1/n^c - 1/n^{c'-2}$, there is exactly one nest with non-zero population by time $r_1 + r_2$. Therefore, with probability at least $1 - 1/n^c - 1/n^{c'-2}$ the house hunting problem is solved in $\mathcal{O}(1/(1-\epsilon)^2)\mathcal{O}(k^3 \log^{1.5} n)$ rounds. □

3.5.3 Composing Algorithm 7 and Density Estimation [83]

We showed that Algorithm 7 is correct and fairly efficient in the weak adversarial model. One way to apply this result is to compose Algorithm 7 with a "subroutine" for population estimation whose guarantees match the assumptions of the weak adversarial model. Next, we briefly describe such a population estimation subroutine [83], referred to as a density estimation algorithm, and state its guarantees. Then, we compose this density estimation algorithm with Algorithm 7 and use Theorem 3.5.3 to conclude that the resulting composition solves the HOUSEHUNTING problem correctly and efficiently.

3.5.3.1 Density Estimation [83]

Consider n ants, positioned uniformly at random at the points of a torus with area A. The density of ants is defined to be $d = n/A$. The density estimation algorithm involves each ant counting how many times it collides with other ants while randomly walking in the grid. The goal is for each ant to compute a good estimate of d based only on the number of such collisions (and the number of random steps it took). Musco et al. [83] show that if each ant walks for t steps and computes the number of collisions x, then the following theorem holds.

Theorem 3.5.21. *After running for t rounds, assuming $t \leq A$, the density estimation algorithm returns $\tilde{d}$ such that:*

1. *$\tilde{d} = x/t$ is an unbiased estimator of d ($\mathbb{E}\left[\tilde{d}\right] = d$),*
2. *For arbitrary ϵ and δ, such that $0 < \epsilon, \delta < 1$, if $t = \Theta\left(\frac{\log(1/\delta)\log\log(1/\delta)\log(1/d\epsilon)}{d\epsilon^2}\right)$, then $\Pr\left[(1-\epsilon)d \leq \tilde{d} \leq (1+\epsilon)d\right] \geq 1-\delta$.*

The above result does not imply that the density estimates of different ants are determined independently from each other. In fact, these estimates are positively correlated because the more two ants collide with each other, the higher both of their density estimates are.

Note that this result also implies that if each ant knows the area A and calculates the density estimate $\tilde{d}$ using the random walking strategy, it can estimate the total number of ants n located in the area/nest. Therefore, assuming each candidate nest is of known area, this algorithm can be used to estimate the population of ants at the nest (the estimate is $\tilde{d} \cdot A$). From a biological perspective, it is reasonable to assume that ants know the approximate area of a nest because they use that information to assess the quality of the nest.

3.5.3.2 Composing Density Estimation and House Hunting

In order to match the guarantees of the density estimation algorithm with the assumptions of the weak adversary, we need to make the following assumptions. For any constants c and c', such that $2 < c' < c$, for any ϵ, such that $0 < \epsilon < 1$, and for any δ such that $0 < \delta < 1$ and $\delta \leq 1/n^{c'}$, consider an instance of the density estimation algorithm with the following properties:

- Density estimation runs for $t = \Theta\left(\frac{\log(1/\delta)\log\log(1/\delta)\log(1/d\epsilon)}{d\epsilon^2}\right)$ rounds[5]. This assumption ensures that density estimation runs for sufficiently many rounds, so that $\Pr\left[\tilde{d} \in [(1-\epsilon)d, (1+\epsilon)d]\right] \geq 1-\delta$.
- The population estimate $A \cdot \tilde{d}$ is at most n. This assumption is not necessarily satisfied in each execution of density estimation, however, we only need it for

[5]These are rounds from the density estimation execution, which we consider separately from the rounds in the house hunting execution. In fact, later we ignore the density estimation rounds.

convenience in composing density estimation with house hunting[6]. Moreover, from a biological perspective, ants are aware of the colony size at any given point, so they would not estimate the population of a nest to be larger than the total colony size.

Suppose that an instance of density estimation, satisfying the assumptions above, runs for each nest in each round of house hunting. That is, in each round and for each each nest n_j with population c_j, the density estimation subroutine returns a collection of values (one for each ant in nest n_j), chosen from the multivariate distribution F_{c_j} in the (ϵ, c', n)-family of distributions. The resulting density estimates are returned to the ants via the calls to the functions **search**(), **go**($\cdot$), and **recruit**($\cdot, \cdot$). The return value $\tilde{d}$ of density estimation corresponds to the return value $\tilde{c_j}$ of these functions. The assumptions above, together with the guarantees of density estimation, ensure that for each nest n_j with population c_j, the return value $(j, \tilde{c_j})$ of the functions satisfies: (1) $\Pr[\tilde{c_j} \in [(1-\epsilon)c_j, (1+\epsilon)c_j]] \geq 1 - 1/n^{c'}$, (2) $\mathbb{E}[\tilde{c_j}] = c_j$, and (3) $\tilde{c_j} \leq n$. These guarantees match the three assumptions of the weak adversarial model (in particular, the definition of the (ϵ, c', n)-family of distributions). Moreover, this composition of density estimation and house hunting ensures that the population estimates of ants in different nests and different rounds are independent because they are the result of independent executions of density estimation. Since the estimates resulting from a single execution of density estimation are not guaranteed to be independent, we assume arbitrary correlation between the estimates of ants in the same nest and the same round.

To compose the density estimation algorithm and Algorithm 7, we ignore the running time of density estimation and assume an instance of it (satisfying the assumptions above) runs at the beginning of each round of the execution of Algorithm 7 for each candidate nest and each ant. The resulting density estimates can then be used as population estimates in the same round of the execution of Algorithm 7.

[6]To compose density estimation and house hunting without this assumption, we can choose some constant upper bounds for ϵ and for each $p(i, r)$, for example, $\epsilon < 1/8$ and $p(i, r) < 3/4$. In this case, the ants' population estimates are guaranteed to be at most n with high probability. Furthermore, once a nest has a population of at least $3n/4$, the recruitment probabilities of the ants in the nest are high enough to ensure convergence within $\mathcal{O}(\log n)$ rounds.

Thus, the following result is a direct corollary of Theorem 3.5.3.

Corollary 3.5.22. *For any constants $2 < c' < c$, and any ϵ, such that $0 < \epsilon, \delta < 1$, with probability at least $1 - 1/n^c - 1/n^{c'-2}$, Algorithm 7, composed with the density estimation algorithm in [83], solves the* HouseHunting *problem in $(2^{22}(c+7)(c+3)k^3 \log^{1.5} n)/(1-\epsilon)^2 = \mathcal{O}(k^3 \log^{1.5} n/(1-\epsilon)^2)$ rounds.*

3.6 Discussion

In the previous section, we discussed the differences between Algorithms 6 and 7 in terms of their resilience to perturbations in the algorithm parameters. Here, we interpret the running times of our two algorithms in terms of efficiency and optimality, and suggest a possible implication of our results.

3.6.1 Running times of our House Hunting Algorithms

First, note that Algorithm 6 is optimal with respect to the model in Section 3.1, as evidenced by the lower bounds in Section 3.2. However, Algorithm 6 exhibits some unnatural and non-ant-like properties, like extreme dependence on synchrony and exact counting of ants at candidate nests. One way to address this mismatch is to weaken our model to not provide ants with the exact number of ants at a nest, as we did in Section 3.5. As we weaken the model, potential extensions of our work would include better lower bounds for the new models.

Next, consider the running time of Algorithm 7. The algorithm performs well in an environment with uncertainty, however, our analysis of its running time is not tight. Although we have no results indicating what the optimal time for house hunting is in the weaker model that includes uncertainty, based on simulations, we know that our analysis of the running time of Algorithm 7 exceeds its actual running time by at least a factor of k. In fact, we believe the running time of Algorithm 7 is $\mathcal{O}(k \log n)$ for the following informal reasons. Consider the populations of ants at nests after the first round and the corresponding recruitment probabilities. Each

nest is expected to receive about n/k ants in expectation, resulting in recruitment probabilities of approximately $1/k$ for each ant. Therefore, the expected time until some nest becomes large (contains a constant fraction of the population) and starts recruiting ants at a constant rate is approximately $\mathcal{O}(k)$. As we saw in the analysis of the lower bound, the $\log n$ factor is usually necessary to ensure that eventually all the ants are located in the same nest. It is also useful in establishing results with high probability. The next section outlines a few modifications to Algorithm 7 to improve its efficiency, at the expense of strengthening the model.

3.6.2 Connection to Population Protocols

Finally, it is worth mentioning a (potentially-surprising) connection between our Algorithm 7 and algorithms for reaching consensus in population protocols. Briefly, in population protocols, each of the n agents interacts with a uniformly selected group of other agents in each round. Various problems have been studied in this setting [8], including consensus and leader election. The typical metrics considered in the analysis of population protocols are time and space complexity, where the space complexity is usually expressed in the number of states each agent needs to encode and execute the algorithm. The *3-majority dynamic* is a consensus protocol in which initially each agent starts with an opinion and the goal is for all agents to converge to the same opinion (not unlike agreeing on a single nest). In each round, each agent interacts with three agents and adopts the majority opinion among them, or an arbitrary opinion if all three are different. The best known analysis of this dynamic has running time of $\mathcal{O}(k^3 \log n)$ [13] rounds until all agents agree on a single opinion. A similar result [11] states that in $\mathcal{O}(k \log n)$ rounds all agents converge to the majority opinion, provided an initial gap of approximately $\Omega(\sqrt{nk \log n})$ votes in favor of the majority opinion.

It is not obvious how to translate between our model and the population protocols model; however, it is interesting to note that both Algorithm 7 and the 3-majority dynamic have the same expected change of a single opinion over a single round: $\mathbb{E}\left[c(i, r+1)\right] = \Theta(c(i,r)(1+c(i,r)-\Sigma(r)))$. Other properties of the random variables

corresponding to the populations of nests are also similar, for example, they are pairwise negatively correlated. We believe our analysis can be used to improve the running time of the 3-majority dynamic from cubic in k to quadratic in k. It remains to see whether a running time of $\mathcal{O}(k \log n)$ rounds is achievable for both algorithms.

3.7 Open Problems

Extensions to the House Hunting Model. For the sake of analysis, we have made many simplifying assumptions about the house-hunting process. We are confident that some of these assumptions can be weakened to make the model more realistic and natural. Some obvious modifications include assuming ants know only an approximation of n, allowing values of k larger than $\mathcal{O}(n/\log n)$, and allowing non-binary nest qualities, variability in the ants' quality sensing, along with some measure of algorithmic performance based on the quality of the chosen nest. Distinguishing between direct transport and tandem runs may also be interesting, paired with a more fine-grained runtime analysis.

Additionally, real ants can only assess nest quality and population approximately. For example, they may estimate nest size (one measure of quality) by randomly walking within the nest and counting how many times they cross their previous path [80]. They seem to estimate nest population by measuring encounter rates with other ants, with a higher encounter rate indicates a higher population at the nest [59, 93]. We already discussed the correctness and performance of the house hunting algorithm under uncertainty in estimating the population of a nest. Adding noisy measurements to the other components of our model and designing algorithms that handle this noise would be a very interesting future direction. It may even be possible to explicitly model lower level behavior and implement subroutines for nest assessment, recruitment, and search which give various runtime and error guarantees.

Extensions to the House Hunting Algorithms. We believe that Algorithm 7 may be a good starting point for work on more realistic house-hunting models.

Below we discuss some interesting possible extensions to the algorithm. Some seem to simply require a more involved analysis, while others seem to require trade-offs in the algorithm's running time and its level of simplicity.

- **Solving house hunting faster:** We conjecture that the $\mathcal{O}(k^3 \log^{1.5} n)$ runtime of Algorithm 7 can be improved to $\mathcal{O}(k \log n)$, which is required because, on average each nest initially contains n/k ants, so ants only recruit with probability $1/k$. Therefore, $\mathcal{O}(k)$ time is required to amplify population gaps by a constant factor. Ideally, ants would all recruit with a probability lower bounded by a constant, but still linearly dependent on the nest populations. This would allow convergence in $\mathcal{O}(\log n)$ rounds. If ants keep track of the round number, they can map this to an estimate $\tilde{k}(r)$ of how many competing nests remain, allowing them to recruit at rate $\mathcal{O}(c(i,r)/n \cdot \tilde{k}(r))$. We believe that such a strategy should yield a relatively natural algorithm converging in $\mathcal{O}(\log^c n)$ rounds.

- **Extend algorithm analysis to population protocols:** As mentioned in Section 3.6, the expected behavior of Algorithm 7 is very close to a well-known population protocol for solving consensus: the *3-majority dynamic*. We conjecture that our analysis extends to the 3-majority dynamic as well, improving its running time by a factor of k. It remains to see if a running time of $\mathcal{O}(k \log n)$ is achievable for both our algorithm and the 3-majority dynamic.

- **Non-binary nest qualities:** Assuming a real-valued nest quality in the range $(0, 1)$ affects the correctness of Algorithm 7 because ants no longer have the notion of a good nest. However, it should be possible to incorporate the quality of the nest into the recruitment probability in order make the algorithm converge to a high-quality nest, without significantly affecting the runtime.

- **Approximate nest assessment and knowledge of n:** Another direction in making the house hunting algorithm closer to real ants is incorporating nest assessment subroutines (similar to density estimation [83]) and using nest qualities in the algorithm. A nest assessment subroutine would provide each ant

with an approximation of the quality of the nest (area, amount of light, etc.) and this value can be incorporated in the probability of recruitment for each ant. An extra challenge in this process would be to provide ants only with approximate information about the size of the colony n.

- **Other uncertainty models:** In Section 3.5, we defined the weak adversarial model with the goal of matching the guarantees of the density estimate algorithm in [83], and thus, being able to compose house hunting with density estimation. However, we can also define various other adversarial models that determine the level of uncertainty in house hunting. For example, one possibility is an adversary that guarantees the population estimates are correct in expectation and are independent for all the ants (unlike the weak adversarial model in which population estimates are allowed to be correlated). Our conjecture is that in this alternative model, the proof and running time of Algorithm 7 remain the same as in the model with no uncertainty, and do not suffer the $1/(1-\epsilon)^2$-factor increase in running time as in the weak adversarial model.

- **Fault tolerance:** Algorithm 7 should support some degree of fault tolerance. A small number of ants suffering from crash-faults or even malicious faults, should not affect the overall populations of recruiting ants and the algorithm's performance. Some similar results in this direction include [13], where the authors analyze the 3-majority dynamic in the presence of adaptive adversarial faults (an agent suffering from a fault changes its opinion arbitrarily). Their results show that the convergence time of the 3-majority dynamic is not affected significantly as long as the number of faults in each round is bounded by $o(\sqrt{n})$.

- **Asynchrony:** Finally, note that Algorithm 7 currently works in synchronous rounds and relies on that assumption to get the correct number of ants at a given nest. However, we believe that, as long as the distribution of ants in candidate nests throughout time stays close to the distribution in the synchronous model, Algorithm 7 can be extended to work in a partially-synchronous model, potentially at the cost of increasing the running time.

Robustness of Randomized Algorithms. Finally, our analysis of Algorithm 7 under uncertainty can serve as the basis of a general study of the robustness properties of (distributed and centralized) randomized algorithms. Our results in Section 3.5 generalized the house hunting model to accommodate perturbations of the probabilities with which ants choose to recruit. Although tolerance to probability perturbations is not a commonly studied robustness property, it can be crucial in the applicability of algorithms to real-world systems, both in engineering and biology.

From our analysis in Section 3.5, and in particular, converting the results with no uncertainty to the weak adversarial model, we notice a few patterns that may help analyze other randomized algorithms under such an adversarial model. Consider a general uncertainty model where each probability p used in a randomized algorithm is instead replaced with an estimate p' that lies in the range $[p(1-\epsilon), p(1+\epsilon)]$ with probability at least $1-\delta$, for some ϵ and δ in $(0,1)$. In this general scenario, consider the properties of the randomized algorithm in comparison to the properties of the algorithm in the original model with no uncertainty.

For statements about the expected values of random variables, we can state a result like Lemma 3.5.0 to show that the expected value in the adversarial model is the same as in the original model plus/minus a small error term that depends on δ. Depending on the exact properties of the randomized algorithm, this error term may get subsumed by other terms in the analysis, as in Lemma 3.5.15 in our case. For stronger probabilistic statements, for example, high-probability statements, we can use the lower or upper bounds (that is, $p(1-\epsilon)$ and $p(1+\epsilon)$, respectively) of the value of p'. This introduces ϵ terms in the resulting properties of the algorithm, and most likely, in the running time of the algorithm. In our case, in Theorem 3.5.3 in Section 3.5, the running time increases by a $1/(1-\epsilon)^2$ factor.

We conjecture that the above patterns for adapting proofs to an uncertainty model may apply to the analysis of a large class of randomized algorithms. Moreover, the degree to which different algorithms tolerate uncertainty can serve as the basis of a new robustness hierarchy of randomized algorithms.

Chapter 4

Task Allocation

In this chapter, we study the task allocation problem in ant colonies, where each ant in the colony needs to choose a task to work on without communicating with other ants and using only limited information from the environment. The results in this chapter include a new model for task allocation with various forms of environment feedback, analysis of three simple task allocation strategies, insight on how these strategies perform under uncertainty, and a discussion on the relevance of our results to real biological systems. This work was done in collaboration with evolutionary biologist Anna Dornhaus who studies the behavior of social insect colonies from both a theoretical and an experimental perspective.

In Section 4.1, we use modeling and analysis techniques from theoretical distributed computing to define an abstract model of task allocation. We define three versions of environment feedback that each ant receives in order to get information about satisfied and unsatisfied tasks. In Section 4.2, we give a few definitions and helper results about the general structure of the task allocation process.

In Section 4.3, we focus on the first type of environment feedback that ants receive. In particular, each ant learns about a uniformly random task in each round. Feedback of this type may be the result of an ant discovering tasks by randomly walking in the nest. We analyze the convergence time of task allocation in this model and conclude that ants require time linear in the number of tasks to allocate correctly, and this value decreases as the ants-to-work ratio increases.

In Section 4.4, we analyze the second type of environment feedback where each ant learns about a uniformly random *unsatisfied* task in each round. In real ant colonies, an ant may discover such a task by randomly walking in the nest and ignoring completed tasks. In this case, we show that task allocation converges much faster, in time logarithmic in the total amount of work needed. We also show that this running time decreases as the logarithm of the ants-to-work ratio increases.

In Section 4.5, we consider environment feedback that informs each ant of a task prioritized by its deficit; in other words, tasks that need more work are likely to be selected more often. A feedback strategy of this type may be the result of a chemical concentration associated with each task that corresponds to the amount of work needed. Again, we show that task allocation converges in time logarithmic in the total amount of work needed and decreases as the ants-to-work ratio increases. Furthermore, we analyze the task allocation process in the presence of uncertainty in the environment feedback.

Finally, in Section 4.6, we provide some numerical examples of our results with parameters chosen from empirical findings about various insect species. We also discuss the implications of our results to the understanding of real insect behavior.

4.1 Model

Let A denote the set of ants and T denote the set of tasks. Each task $i \in T$ has an integer demand d_i that represents the minimum number of ants required to work on task i in order to satisfy the task. Let w_i denote the total number of ant units of work currently supplied to task i. Let $\vec{w}$ and $\vec{d}$ denote the vectors of w_i and d_i values, respectively, for each $1 \leq i \leq |T|$. The $\vec{d}$ vector is static, while $\vec{w}$ changes over time depending on the different tasks ants choose to work on.

Clearly, in order for all demands to be met, there should be sufficiently many ants in the colony. We assume that $|A| \geq c \cdot \sum_{i \in T} d_i$ for $c \geq 1$.

The structure of the task allocation system we consider is illustrated in Figure 4-1. The environment component contains state information about all tasks, their demands

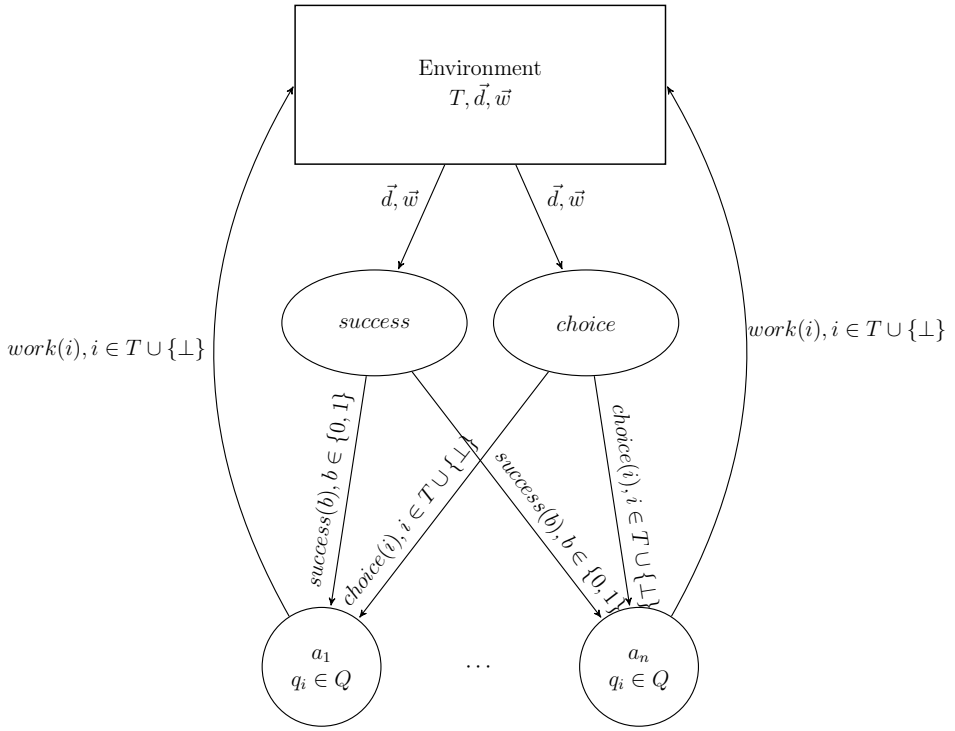

Figure 4-1: The task allocation system consists of an environment component, the *success* and *choice* feedback components, and n ant components.

and the work supplied to each task at the current point in time. The environment component informs the *success* and *choice* feedback components of the demands and the work provided to tasks. The *success* component provides each ant component with information $success(b), b \in \{0, 1\}$ about the success of the ant at the task i it is currently working on. The *choice* component provides each ant component with information $choice(i), i \in T \cup \{\perp\}$ about some (alternative) task i, where $\perp$ indicates "no task". Finally, each ant component updates the environment with the work it has completed on task $i \in T \cup \{\perp\}$ through $work(i)$. An ant component output $work(\perp)$ indicates that the ant did not complete any work.

Environment: The environment consists of the set of tasks, their demands and the number of ants currently working on each task. The environment outputs the $\vec{d}$ and $\vec{w}$ vectors to the *success* and *choice* components, providing them with information about the work supplied to tasks and their demands. The input to the environment component is the work that each ant provides to each task denoted by $work(i), i \in$

$T \cup \{\perp\}$. As a result, the environment updates the $\vec{w}$ vector accordingly.

Ants: Each ant $a \in A$ has a state $q \in Q = \{q_\perp, q_1, q_2, \cdots, q_{|T|}\}$ at each point in time, where $q_\perp$ indicates that ant a is not working on any task and each state q_i, for $i \in \{1, \cdots, |T|\}$, indicates that ant a is working on task i. Each ant is modeled as a finite state machine with transition function $\delta : Q \times (\{0,1\} \times (T \cup \{\perp\})) \to Q$; in other words, each ant's new state is determined by its old state and its inputs from the *success* and *choice* components. Let q be the current state of some ant a, and let q' be the resulting state of ant a after applying δ. In each step, q' is determined as follows: $q' = q$ if *success* outputs 1, and $q' = q_i$ if *success* outputs 0 and *choice* outputs $i \in T \cup \{\perp\}$. The new state q' of the ant directly determines its output to the environment component.

Feedback: The environment feedback components *success* and *choice* provide each ant with a boolean and a task in $T \cup \{\perp\}$, determined based on $\vec{w}$ and $\vec{d}$. The input to the feedback components is the $\vec{d}$ and $\vec{w}$ vectors. The output of *success* is a boolean $\{0,1\}$, and the output of *choice* is some task in $T \cup \{\perp\}$. Since randomness usually plays a role in the environment, the output values of *success* and *choice* may be determined randomly.

Execution: The execution of any algorithm solving the task allocation problem starts at time 0 and proceeds in synchronous rounds, such that each round $r+1$, for $r \geq 0$, denotes the transition from time r to time $r+1$. In each round $r+1$, the environment component provides outputs to *success* and *choice* containing the demand and work vectors $\vec{d}$ and $\vec{w(r)}$. The *success* and *choice* components provide each ant component with a boolean ($success(b), b \in \{0,1\}$) and a task ($choice(i), i \in T \cup \{\perp\}$) output. Each ant component performs a state transition using its δ transition function and updates the environment component with the work ($work(i), i \in T \cup \{\perp\}$) it performed. An execution α is a sequence of alternating (1) states of the environment, and mappings from ants to (2) *success* outputs, (3) *choice* outputs, and (4) work inputs. Formally, $\alpha = (Q_0, S_1, C_1, W_1, Q_1, S_2, C_2, W_2, \cdots)$, where, for

each $r \geq 0$, Q_r is the state of the environment at time r, S_{r+1} is a mapping of type $A \to \{0, 1\}$, C_{r+1} is a mapping of type $A \to T \cup \{\perp\}$, and W_{r+1} is a mapping of type $A \to T \cup \{\perp\}$. The S_{r+1} mapping refers to the *success* outputs to all ants in round $r + 1$, the C_{r+1} mapping refers to the *choice* outputs to all ants in round $r + 1$, and W_{r+1} mapping refers to the inputs to the environment from all ants in round $r + 1$.

Problem Statement: A state s_r of the environment component at time $r \geq 0$ *satisfies* some task $i \in T$ if $d_i \leq w_i(r)$. An execution α satisfies all tasks if there exists a time $r \geq 0$ such that for each $r' \geq r$, state $s_{r'}$ of the environment satisfies task i for all $i \in T$. A probabilistic execution α satisfies all tasks with probability $1 - \delta$, for any $0 < \delta < 1$, if, with probability at least $1 - \delta$, there exists a time $r \geq 0$ such that for each $r' \geq r$, state $s_{r'}$ of the environment satisfies task i for all $i \in T$.

The specification of *success* and some of the specifications of *choice* in this section are inspired by the biological model by Pacala et al. [86] and simplified for the sake of easier analysis.

Success Component The first component, *success*, determines whether each ant is successful at the task it is currently working on. We consider *success* components that satisfy the following conditions in each execution and at each time r of the execution. Let $S(i, r)$ be the following set:

$$\{a \mid a \text{ is in state } q_i \text{ at time } r \text{ and receives } success(1) \text{ in round } r + 1\}.$$

For each task $i \in T$, $|S(i, r)| = \min(d_i, w_i(r))$. Also, each ant in state $q_\perp$ at time r receives $success(0)$ in round $r+1$. The *success* component provides information that allows excess ants working on a satisfied task to switch to another task.

Choice Component The *choice* component returns a candidate task to each ant as an alternative task to work on. We consider three different specifications of *choice*:

1. *choice* returns a task drawn from all the tasks in T uniformly at random.

2. *choice* returns a task drawn from the set of unsatisfied tasks, $U(r) = \{i \mid d_i > w_i(r)\}$, uniformly at random. If there is no such task, then *choice* returns $\perp$.

3. *choice* returns a task i drawn from the set of all unsatisfied tasks with probability $(d_i - w_i(r))/\sum_{j \in U(r)}(d_j - w_j(r))$. This option corresponds to the scenario where ants can somehow sense the need to work on each task, and are more likely to work on tasks with high deficit $d_i - w_i(r)$ compared to the total deficit of all unsatisfied tasks $\sum_{j \in U(r)}(d_j - w_j(r))$.

4.2 General Definitions and Lemmas

In this section, we give some basic definitions and results that will be used in the subsequent analyses of the convergence times for the various *choice* options.

Definitions. All definitions are with respect to a fixed execution α. For each task $i \in T \cup \{\perp\}$ and each time r, let $A_i(r)$ denote the set of ants in state q_i at time r. A task is *satisfied* at time r if $d_i \leq w_i(r)$. Let $S(r)$ denote the set of satisfied tasks at time r. Let $U(r) = T \setminus S(r)$ denote the set of unsatisfied tasks at time r.

We begin by showing some basic properties of the *success* component. These results hold regardless of the particular *choice* component specification. Consider an arbitrary execution α of the task allocation system with the *success* component and any one of the three *choice* components.

Lemma 4.2.1. *For each task $i \in T$, each time r, each time r', such that $r' \geq r$, and each $d \in \mathbb{N}$, such that $d \leq d_i$, if $w_i(r) \geq d$, then $w_i(r') \geq d$.*

Proof. Consider fixed task $i \in T$, time r, and value $d \leq d_i$, and suppose that $w_i(r) \geq d$. The proof is by induction on r' for $r' \geq r$. In the base case $r' = r$, and by assumption $w_i(r) \geq d$. Suppose in some round $r' > r$ it is true that $w_i(r') \geq d$. We need to show that $w_i(r'+1) \geq d$. By the definition of *success* applied to round $r'+1$, the number of the ants working on task i at time r' that receive 1 from *success* in round $r'+1$ is $\min\{w_i(r'), d_i\} \geq d$. By the definition of the transition function δ, an

ant that receives *success*(1) in round $r' + 1$ keeps working on its current task (task i) at time $r' + 1$, so $w_i(r' + 1) \geq d$. □

Corollary 4.2.2. *For each time r, $|U(r)| \geq |U(r + 1)|$ and $|S(r)| \leq |S(r + 1)|$.*

For each task $i \in T$ and each time r, let $\Phi_i(r) = \max\{0, (d_i - w_i(r))\}$ be the *deficit* of task i at time r. If $i \in U(r)$, then $\Phi_i(r) = d_i - w_i(r)$. We define the *total deficit* at time r:

$$\Phi(r) = \sum_{i \in T} \Phi_i(r).$$

Lemma 4.2.3. *For each time r and each task $i \in T$, $\Phi_i(r) \geq \Phi_i(r + 1)$.*

Proof. Consider fixed task $i \in T$ and time r. If $w_i(r) \geq d_i$, then by Lemma 4.2.1, $w_i(r + 1) \geq d_i$, so $\Phi_i(r) = \Phi_i(r + 1) = 0$. Otherwise, by Lemma 4.2.1, $w_i(r + 1) \geq w_i(r)$, so $\Phi_i(r) = d_i - w_i(r) \geq \max\{0, d_i - w_i(r + 1)\} = \Phi_i(r + 1)$. In either case, $\Phi_i(r) \geq \Phi_i(r + 1)$. □

Corollary 4.2.4. *For each time r, $\Phi(r) \geq \Phi(r + 1)$.*

Define an ant to be *inactive* in round r, for $r > 0$, if it is in state $q_\perp$ at time $r - 1$ or if it receives *success*(0) in round r. In other words, an ant is inactive if it is not working on any task, or if it unsuccessful at the current task it is working on. So, the number of inactive ants in some round $r + 1$ is $|A_\perp(r)| + \sum_{i \in S(r)}(w_i(r) - d_i)$.

Recall that $|A| \geq c \cdot \sum_{i \in T} d_i$ for $c \geq 1$.

Lemma 4.2.5. *The number of inactive ants in round $r + 1$ is at least $c \cdot \Phi(r)$.*

Proof. Similarly to the proof of Lemma 4.2.5, the total number of ants $|A|$ can be decomposed into the number of ants not working on any task, the number of ants working on satisfied tasks, and the number of ants working on unsatisfied tasks:

$$|A| = \sum_{i \in U(r)} w_i(r) + \sum_{i \in S(r)} w_i(r) + |A_\perp(r)|.$$

Based on the assumption that $|A| \geq c \cdot \sum_{i \in T} d_i$, we know that $|A|/c \geq \sum_{i \in S(r)} d_i + \sum_{i \in U(r)} d_i$. Also, by the definition of $\Phi(r)$, it is true that $\Phi(r) \leq \sum_{i \in T} d_i \leq |A|/c$. The number of inactive ants in round $r+1$ is:

$$\begin{aligned}
|A(r)_\perp| + \sum_{i \in S(r)} (w_i(r) - d_i) &= |A| - \sum_{i \in U(r)} w_i(r) - \sum_{i \in S(r)} w_i(r) + \sum_{i \in S(r)} w_i(r) - \sum_{i \in S(r)} d_i \\
&= |A| - \sum_{i \in U(r)} w_i(r) - \sum_{i \in S(r)} d_i \\
&\geq |A| - \sum_{i \in U(r)} w_i(r) - \left(\frac{|A|}{c} - \sum_{i \in U(r)} d_i \right) \\
&= |A| \left(\frac{c-1}{c} \right) + \sum_{i \in U(r)} (d_i - w_i(r)) \\
&\geq (c-1)\Phi(r) + \Phi(r) = c \cdot \Phi(r).
\end{aligned}$$

□

Next, we show a simple lemma that will be useful in analyzing *choice* components that always provide ants with an unsatisfied task (options (2) and (3)).

Lemma 4.2.6. *Suppose that in each round $r+1$ such that $U(r) \neq \emptyset$, choice returns a task $i \in U(r)$ to each ant. Then, all tasks are satisfied by time T.*

Proof. By Corollary 4.2.2, $|U(r)| \geq |U(r+1)|$ for any r. Therefore, it suffices to show that if $U(r) \neq \emptyset$, then $|U(r)| > |U(r+1)|$. Assume to the contrary that for some time r, such that $|U(r)| \neq 0$, $|U(r)| = |U(r+1)|$. By the definition of an unsatisfied task, $w_i(r+1) < d_i$ for each $i \in U(r)$. By Lemma 4.2.5, the number of inactive ants in round $r+1$ is at least $\Phi(r)$, and, by assumption, *choice* returns an unsatisfied task to each inactive ant. We have the following contradiction:

$$\begin{aligned}
\Phi(r) &\leq \sum_{i \in U(r)} (w_i(r+1) - w_i(r)) && \text{by the specification of } \textit{choice}, \\
&< \sum_{i \in U(r)} (d_i - w_i(r)) && \text{since } w_i(r+1) \leq d_i \text{ for each } i \in U(r), \\
&= \Phi(r) && \text{by the definition of } \Phi(r).
\end{aligned}$$

□

The following lemma lets us bound the expected values of the total deficit and the number of unsatisfied tasks given that the probability of satisfying each task is bounded from below.

Lemma 4.2.7. *Suppose that for each unsatisfied task $i \in U(r)$ it is true that* $\Pr[w_i(r+1) \geq d_i] \geq p$. *Then,* $\mathbb{E}[|U(r+1)|] \leq |U(r)| \cdot (1-p)$ *and* $\mathbb{E}[\Phi(r+1)] \leq \Phi(r) \cdot (1-p)$.

Proof. The expected number of unsatisfied tasks at time $r+1$ is:

$$\begin{aligned}
\mathbb{E}\left[|U(r+1)|\right] &\leq |U(r)| - \sum_{i \in U(r)} \Pr[w_i(r+1) \geq d_i] \\
&\leq |U(r)| - |U(r)| \cdot p \\
&\leq |U(r)| \cdot (1-p).
\end{aligned}$$

By assumption, $\Pr[\Phi_i(r+1) \leq 0] = \Pr[d_i - w_i(r+1) \leq 0] \geq p$. Therefore:

$$\begin{aligned}
\mathbb{E}[\Phi(r+1)] &= \sum_{i \in T} \mathbb{E}[\Phi_i(r+1)] \\
&= \sum_{i \in T} \mathbb{E}[\Phi_i(r+1) \mid \Phi_i(r+1) \leq 0] \cdot \Pr[\Phi_i(r+1) \leq 0] \\
&\quad + \mathbb{E}[\Phi_i(r+1) \mid \Phi_i(r+1) > 0] \cdot \Pr[\Phi_i(r+1) > 0] \\
&\leq \mathbb{E}[\Phi_i(r+1) \mid \Phi_i(r+1) > 0] \cdot \Pr[\Phi_i(r+1) > 0] && \Phi_i(r+1) \geq 0 \\
&\leq \Phi_i(r) \cdot (1-p) && \Phi_i(r+1) \leq \Phi_i(r).
\end{aligned}$$

□

Next, we analyze the three variations of the *choice* component. In Section 4.3, we analyze the convergence time of task allocation when *choice* returns a uniformly random task, in Section 4.4, we analyze the convergence time of task allocation when *choice* returns a uniformly random *unsatisfied* task, and in Section 4.5, we analyze the convergence time of task allocation when *choice* returns an unsatisfied task with probability proportional to its deficit.

4.3 Uniformly Random Tasks

In this section, we consider the first option for the *choice* component, where in each round *choice* returns a task i with probability $1/|T|$.

For Lemma 4.3.1, 4.3.2 and 4.3.3, assume α is a fixed execution and $r \geq 0$ is some fixed time in α. We consider the state variables at time r and the outputs of *success* in round $r+1$ to be fixed, and we consider the probability distribution over the randomness introduced by the *choice* outputs in round $r+1$.

By Lemma 4.2.5, we know that the number of inactive ants in round $r+1$ is at least $c \cdot \Phi(r)$. By the definition of *choice* in this section, each inactive ant starts working on each task i with probability $1/|T|$. In the next lemma, we show that, in each round, the expected number of new ants to join each unsatisfied task is at least $c \cdot \Phi(r)/|T|$.

Lemma 4.3.1. *For each task $i \in U(r)$, $\mathbb{E}[w_i(r+1) - w_i(r)] \geq c \cdot \Phi(r)/|T|$.*

Proof. Note that, in the expression we want to prove, $w_i(r+1)$ is a random variable, whereas $\Phi(r)$ and $w_i(r)$ are fixed values. By Lemma 4.2.5, the number of inactive ants in round $r+1$ is at least $c \cdot \Phi(r)$. Therefore, for each task i, the expected number of ants that are inactive in round $r+1$ and working on task i at time $r+1$ is:

$$\mathbb{E}[w_i(r+1) - w_i(r)] \geq \frac{c \cdot \Phi(r)}{|T|}.$$

□

After some ants join task i in round $r+1$, it is not guaranteed that the entire new set of ants remains working on task i because some ants may be unsuccessful if task i does not require that many workers. Assuming $c \leq |T|$, since the total deficit is $\Phi(r)$ and there are $|T|$ tasks, we show that the sum of deficits of the top c tasks is at least $c \cdot \Phi(r)/|T|$ (which can be 0 if all tasks are satisfied). Therefore, in expectation, at least $c \cdot \Phi(r)/|T|$ of the new ants that join these tasks will remain working on them. In the next lemma, we show that the expected total deficit $\Phi(r)$ decreases by approximately $c \cdot \Phi(r)/|T|$ in round $r+1$.

Lemma 4.3.2. *For $c \leq |T|$, $\mathbb{E}[\Phi(r+1)] \leq (1 - c/4|T|)\Phi(r)$.*

Proof. The expected decrease in one round of the value of $\Phi(r)$ is:

$$\begin{aligned}
\mathbb{E}[\Phi(r) - \Phi(r+1)] &\geq \sum_{i \in T} \mathbb{E}[\Phi_i(r) - \Phi_i(r+1)] && \text{by Lemma 4.2.3,} \\
&= \sum_{i \in T} \mathbb{E}\left[\min\left\{\left(w_i(r+1) - w_i(r)\right), \left(d_i - w_i(r)\right)\right\}\right] \\
&\geq \frac{1}{4} \sum_{i \in T} \min\left\{\mathbb{E}\left[w_i(r+1) - w_i(r)\right], \left(d_i - w_i(r)\right)\right\} \\
&\geq \frac{1}{4} \sum_{i \in T} \min\left\{\frac{c \cdot \Phi(r)}{|T|}, \left(d_i - w_i(r)\right)\right\} && \text{by Lemma 4.3.1.}
\end{aligned}$$

The fourth inequality above is derived by applying Corollary A.1.2 is to random variable $X = w_i(r+1) - w_i(r)$ and the fixed value $d_i - w_i(r) \geq 1$. Random variable X can be expressed as a sum of k independent binary random variables, where k is the number of inactive ants in round $r+1$ and each ant contributes 1 to the sum if it receives *choice*(i) in round r, and 0 otherwise.

Suppose in contradiction that:

$$\sum_{i \in T} \min\left\{\frac{c \cdot \Phi(r)}{|T|}, \left(d_i - w_i(r)\right)\right\} < \frac{c \cdot \Phi(r)}{|T|}.$$

It must be the case that for each $i \in T$, $d_i - w_i(r) < c \cdot \Phi(r)/|T|$. Since $c \leq |T|$, $\sum_{i \in T}(d_i - w_i(r)) < \Phi(r)$, a contradiction.

We have $\mathbb{E}[\Phi(r) - \Phi(r+1)] \geq (1/4)(c \cdot \Phi(r)/|T|)$, so $\mathbb{E}[\Phi(r+1)] \leq (1 - c/4|T|)\Phi(r)$. □

Next, we consider the case of $c > |T|$. We can express c as a multiple of $|T|$: $c = c' \cdot |T|$ for some $c' > 1$. Note that c' is not necessarily a constant. We show that in each round, the probability to satisfy each task is at least some constant, and consequently, the expected number of unsatisfied tasks decreases by a constant fraction in each round.

Lemma 4.3.3. *For $c > |T|$, $\mathbb{E}[|U(r+1)|] \leq |U(r)|e^{-c'(1-1/c')^2/2}$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(r)e^{-c'(1-1/c')^2/2}$.*

Proof. By Lemma 4.2.5, the number of inactive ants in round $r+1$ is at least $c \cdot \Phi(r)$. Therefore, for each $i \in U(r)$, $\mathbb{E}[w_i(r+1) - w_i(r)] \geq c \cdot \Phi(r)/|T| = c' \cdot \Phi(r)$. By a Chernoff bound it follows that:

$$\begin{aligned}
&\Pr\left[(w_i(r+1) - w_i(r)) < \Phi_i(r)\right] \\
&\quad \leq \Pr\left[(w_i(r+1) - w_i(r)) < \Phi(r)\right] \\
&\quad \leq \Pr\left[(w_i(r+1) - w_i(r)) < \left(\frac{1}{c'}\right)\mathbb{E}\left[w_i(r+1) - w_i(r)\right]\right] \\
&\quad \leq e^{-\frac{\mathbb{E}[w_i(r+1)-w_i(r)]\left(1-\frac{1}{c'}\right)^2}{2}} \\
&\quad \leq e^{-\frac{c'\cdot\Phi(r)\left(1-\frac{1}{c'}\right)^2}{2|U(r)|}} && \text{since } \Phi(r) \geq 1, \\
&\quad \leq e^{-\frac{c'\left(1-\frac{1}{c'}\right)^2}{2}}.
\end{aligned}$$

Therefore, $\Pr[w_i(r+1) \geq d_i] \geq 1 - e^{-c'(1-1/c')^2/2}$, so by Lemma 4.2.7, it follows that $\mathbb{E}[|U(r+1)|] \leq |U(r)| \cdot e^{-c'(1-1/c')^2/2}$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(r) \cdot e^{-c'(1-1/c')^2/2}$. □

Finally, we fix some arbitrary deterministic *success* components in each round, and we analyze the total running time of task allocation for an arbitrary probabilistic execution of the resulting system. In the next theorem, we start at time 0, when the total deficit is $\Phi(0)$, and inductively apply Lemmas 4.3.2 and 4.3.3 and iterated expectation.

Theorem 4.3.4. *For $c \leq |T|$ and for any δ, $0 < \delta < 1$, with probability at least $1-\delta$, all tasks are satisfied by time $(4/c)|T|(\ln \Phi(0) + \ln(1/\delta))$.*

Proof. First, we show by induction that for each $r \geq 0$ the following holds at time r:

$$\mathbb{E}[\Phi(r)] \leq \Phi(0)\left(1 - \frac{c}{4|T|}\right)^r. \tag{4.1}$$

In the base case, $r = 0$. By Corollary 4.2.4, $\Phi(r) \leq \Phi(0)$.

For the inductive step, we assume the statement is true for some fixed $r \geq 0$, and we show it is true for $r+1$. First, consider a fixed prefix α of the execution of length r and some fixed outputs of the *success* component in round $r+1$. Recall

that by assumption, the outputs to all ants of *success* are determined by an arbitrary deterministic component. By Lemma 4.3.2, $\mathbb{E}[\Phi(r+1)] \leq \Phi(r)\,(1 - c/4|T|)$. Note that this expectation has a different meaning from the one in Equation (4.1) that we want to show. Here, the expectation is taken only over the randomness induced by the *choice* component outputs in round $r+1$ (starting from a fixed time at the end of the execution prefix α), and $\Phi(r)$ is a *fixed value*. Next, we need to state a bound on $\mathbb{E}[\Phi(r+1)]$ over all random choices from the beginning of the probabilistic execution up to time r in terms of the *random variable* $\Phi(r)$. Since all prefixes of length r are disjoint, by the law of total expectation, for all prefixes of length r in the probabilistic execution, it is true that $\mathbb{E}[\Phi(r+1) \mid \Phi(r)] \leq \Phi(r)\,(1 - c/4|T|)$.

Now we use the law of iterated expectation and the inductive hypothesis to obtain a bound on $\mathbb{E}[\Phi(r+1)]$.

$$\begin{aligned}
\mathbb{E}[\Phi(r+1)] &= \mathbb{E}[\mathbb{E}[\Phi(r+1) \mid \Phi(r)]] && \text{by iterated expectation,} \\
&\leq \mathbb{E}\left[\Phi(r)\left(1 - \frac{c}{4|T|}\right)\right] && \text{by the above bound,} \\
&= \mathbb{E}[\Phi(r)]\left(1 - \frac{c}{4|T|}\right) && \text{by linearity of expectation,} \\
&\leq \Phi(0)\left(1 - \frac{c}{4|T|}\right)^{r+1} && \text{by the inductive hypothesis.}
\end{aligned}$$

Thus, we have shown that, for every $r \geq 0$, $\mathbb{E}[\Phi(r)] \leq \Phi(0)\,(1 - c/4|T|)^r$.

For $r = (4/c)|T|(\ln \Phi(0) + \ln(1/\delta))$ we get:

$$\begin{aligned}
\mathbb{E}[\Phi(r)] &\leq \Phi(0)\left(1 - \frac{c}{4|T|}\right)^r \\
&\leq \Phi(0) \cdot e^{-\frac{4c|T|(\ln \Phi(0) + \ln(1/\delta))}{4c|T|)}} && \text{by } (1-a)^b \leq e^{-ab} \text{ for } a, b \in \mathbb{R} \text{ and } a, b > 0, \\
&\leq \Phi(0) \cdot e^{-(\ln \Phi(0) + \ln(1/\delta))} \\
&\leq \Phi(0)\left(\frac{1}{\Phi(0)}\right)\delta \leq \delta.
\end{aligned}$$

By a Markov bound, we get $\Pr[\Phi(r) \geq (1/\delta)\mathbb{E}[\Phi(r)]] \leq \delta$. Therefore, with probability at least $1-\delta$, the total deficit is strictly less than 1 by time $(4/c)|T|(\ln \Phi(0) + \ln(1/\delta))$.

This implies that all tasks are satisfied by that time. □

Corollary 4.3.5. *For $c \leq |T|$ and for any δ and ϵ, $0 < \delta, \epsilon < 1$, with probability at least $1 - \delta$, the deficit at time $(4/c)|T|(\ln(1/\epsilon) + \ln(1/\delta))$ is at most $\epsilon \cdot \Phi(0)$.*

The proof is similar to the proof of Theorem 4.3.4; we show that $\mathbb{E}[\Phi(r+1)] \leq \Phi(0) \cdot \epsilon \cdot \delta$ instead of $\mathbb{E}[\Phi(r+1)] \leq \delta$.

Theorem 4.3.6. *For $c > |T|$ and for any δ, $0 < \delta < 1$, with probability at least $1-\delta$, all tasks are satisfied by time $(2/(c'(1-1/c')^2))(\min\{\ln|T|, \ln\Phi(0)\} + \ln(1/\delta))$.*

Proof. Initially, $|U(0)| \leq |T|$. By Lemma 4.3.3, $\mathbb{E}[|U(r+1)|] \leq |U(r)| \cdot e^{-\frac{c'\left(1-\frac{1}{c'}\right)^2}{2}}$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(r) \cdot e^{-\frac{c'\left(1-\frac{1}{c'}\right)^2}{2}}$. Similarly to Theorem 4.3.4, we can inductively apply Lemma 4.3.3 and iterated expectation to show that for each $r \geq 0$, it is true that $\mathbb{E}[|U(r+1)|] \leq |T| \cdot e^{-rc'(1-1/c')^2/2}$ and $\mathbb{E}[\Phi(r+1)|] \leq \Phi(0) \cdot e^{-rc'(1-1/c')^2/2}$. For $r = (2/(c'(1-1/c')^2))(\min\{\ln|T| + \ln\Phi(0)\} + \ln(1/\delta))$ at least one of the following is true:

$$\mathbb{E}[|U(r)|] \leq |T| \cdot e^{-(2/(c'(1-1/c')^2))(\ln|T|+\ln(1/\delta))} \leq \delta$$
$$\mathbb{E}[\Phi(r)] \leq \Phi(0) \cdot e^{-(2/(c'(1-1/c')^2))(\ln\Phi(0)+\ln(1/\delta))} \leq \delta.$$

Therefore, by a Markov bound, with probability at least $1-\delta$, $|U(r)| < 1$ or $\Phi(r) < 1$, implying that all tasks are satisfied by time $(2/(c'(1-1/c')^2))(\min\{\ln|T|, \ln\Phi(0)\} + \ln(1/\delta))$. □

Corollary 4.3.7. *For $c > |T|$ and for any δ and ϵ, $0 < \delta, \epsilon < 1$, with probability at least $1-\delta$, the deficit at time $(2/(c'(1-1/c')^2))(\ln(1/\epsilon) + \ln(1/\delta))$ is at most $\epsilon \cdot \Phi(0)$.*

The proof is similar to the proof of Theorem 4.3.6; we ignore the case when $|T| < \Phi(0)$ and we show that $\mathbb{E}[\Phi(r+1)] \leq \Phi(0) \cdot \epsilon \cdot \delta$ instead of $\mathbb{E}[\Phi(r+1)] \leq \delta$.

We can combine the results of Theorems 4.3.4 and 4.3.6 by slightly weakening Theorem 4.3.6. Clearly, if c' is extremely close to 1 (that is, c is extremely close to $|T|$), the $1/(c'(1-1/c')^2)$ term in Theorem 4.3.6 becomes very large, and in the limit the running time becomes ∞. Therefore, we can take the result of Theorem 4.3.4 for

$c \leq 2|T|$ and the result of Theorem 4.3.6 for $c > 2|T|$, in which case $c' > 2$ and the $2/(c'(1-1/c')^2)$ can be bounded by 4.

Corollary 4.3.8. *For any δ, $0 < \delta < 1$, with probability at least $1-\delta$, all tasks are satisfied by time $\mathcal{O}(c^{-1}) \cdot \mathcal{O}(|T|(\ln \Phi(0) + \ln(1/\delta)))$.*

Similarly, we can combine Corollaries 4.3.5 and 4.3.7.

Corollary 4.3.9. *For any δ and ϵ, $0 < \delta, \epsilon < 1$, with probability at least $1-\delta$, the deficit at time $\mathcal{O}(c^{-1}) \cdot \mathcal{O}(|T|(\ln(1/\epsilon) + \ln(1/\delta)))$ is at most $\epsilon \cdot \Phi(0)$.*

4.4 Uniformly Random Unsatisfied Tasks

Here, we consider the second option for the *choice* component, defined in Section 4.1, where in each round *choice* returns a task $i \in U(r)$ with probability $1/|U(r)|$.

The results in this section follow the same structure as the results in Section 4.3. In this section, we do not include as much detail in some of the results. For example, in the proof of Theorem 4.4.5, we skip the detailed application of law of total expectation and iterated expectation since they follow the same pattern as in the proof of Theorem 4.3.4.

For Lemma 4.4.1 and 4.4.2, assume α is a fixed execution and $r \geq 0$ is some fixed time in α. We consider the state variables at time r and the outputs of *success* in round $r+1$ to be fixed, and we consider the probability distribution over the randomness introduced by the *choice* outputs in round $r+1$.

Lemma 4.4.1. *For each unsatisfied task $i \in U(r)$, $\mathbb{E}[w_i(r+1) - w_i(r)] \geq \Phi(r)/|U(r)|$.*

Proof. By Lemma 4.2.5, the number of inactive ants in round $r+1$ is at least $\Phi(r)$. Since the number of unsatisfied tasks is $|U(r)|$, for each unsatisfied task i, it is true that $\mathbb{E}[w_i(r+1) - w_i(r)] \geq \Phi(r)/|U(r)|$. □

We show that in round $r+1$ at least one of the following happens: (1) the total deficit decreases by a constant fraction, or (2) the number of unsatisfied tasks decreases by a constant fraction. To show the first property, we consider tasks with

a fairly high deficit, which are not likely to get satisfied in one round. We show that the number of new ants joining such tasks is enough to decrease the total deficit by a constant fraction. To show the second property (the number of unsatisfied tasks decreases by a constant fraction), we focus on tasks with fairly low deficit which are likely to get satisfied within one round. We show that these tasks are enough to decrease the total number of unsatisfied tasks by a constant fraction in one round.

Lemma 4.4.2. *For $c \geq 1$, at least one of the following is true:*

$$\mathbb{E}[\Phi(r+1)] \leq \left(\frac{15}{16}\right)\Phi(r) \tag{4.2}$$

$$\mathbb{E}[|U(r+1)|] \leq \left(1 - \frac{1-e^{-1/8}}{2}\right)|U(r)| \tag{4.3}$$

Proof. We define task $i \in U(r)$ to have a *high deficit* if $\Phi_i(r) > \Phi(r)/(2|U(r)|)$. Since the deficit is always an integer, $\Phi_i(r) \geq \lceil \Phi(r)/(2|U(r)|) \rceil$. Otherwise, if $\Phi_i(r) \leq \Phi(r)/(2|U(r)|)$ for a task $i \in U(r)$, task i is defined to have a *low deficit*. Let $H(r)$ be the set of high-deficit tasks and let $L(r) = U(r) \setminus H(r)$ be the set of low-deficit tasks. We consider two cases based on the value of $|H(r)|$.

Case 1: $|H(r)| \geq |U(r)|/2$. The expected decrease of $\Phi(r)$ after one round is:

$$\begin{aligned}
\mathbb{E}[\Phi(r) - \Phi(r+1)] &\geq \sum_{i \in H(r)} \mathbb{E}[\Phi_i(r) - \Phi_i(r+1)] && \text{by Lemma 4.2.3,} \\
&= \sum_{i \in H(r)} \mathbb{E}\left[\min\left\{(w_i(r+1) - w_i(r)), (d_i - w_i(r))\right\}\right] \\
&\geq \sum_{i \in H(r)} \mathbb{E}\left[\min\left\{(w_i(r+1) - w_i(r)), \left\lceil \frac{\Phi(r)}{2|U(r)|} \right\rceil\right\}\right] \\
&\geq \sum_{j \in H(r)} \frac{1}{4} \min\left\{\mathbb{E}\left[w_i(r+1) - w_i(r)\right], \left\lceil \frac{\Phi(r)}{2|U(r)|} \right\rceil\right\} \\
&\geq \sum_{j \in H(r)} \frac{1}{4} \min\left\{\frac{\Phi(r)}{|U(r)|}, \left\lceil \frac{\Phi(r)}{2|U(r)|} \right\rceil\right\} && \text{by Lemma 4.4.1,} \\
&= \frac{|H(r)|}{4} \cdot \frac{\Phi(r)}{2|U(r)|} \\
&\geq \frac{|U(r)|}{8} \cdot \frac{\Phi(r)}{2|U(r)|} \geq \frac{\Phi(r)}{16}.
\end{aligned}$$

In this case, Equation (4.2) is satisfied.

Case 2: $|H(r)| < |U(r)|/2$, so $|L(r)| \geq |U(r)|/2$. In this case, many of the tasks have low deficit and each one of them is fairly close to being satisfied.

Consider a low-deficit task $i \in L(r)$. By Lemma 4.4.1, $\mathbb{E}\left[w_i(r+1) - w_i(r)\right] \geq \Phi(r)/|U(r)|$. Also, by definition, $\Phi(r) \geq |U(r)|$, so $\mathbb{E}\left[w_i(r+1) - w_i(r)\right] \geq 1$. Applying a Chernoff bound, we get:

$$\begin{aligned}
&\Pr\left[(w_i(r+1) - w_i(r)) < \frac{\Phi(r)}{2|U(r)|}\right] \\
&\quad \leq \Pr\left[(w_i(r+1) - w_i(r)) < \left(\frac{1}{2}\right)\mathbb{E}\left[w_i(r+1) - w_i(r)\right]\right] \\
&\quad \leq e^{-\mathbb{E}[w_i(r+1)-w_i(r)]/8} && \text{since } \mathbb{E}[w_i(r+1) - w_i(r)] \geq 1, \\
&\quad \leq e^{-1/8}.
\end{aligned}$$

Thus, the probability task i is satisfied at time $r+1$ is:

$$\begin{aligned}
\Pr\left[w_i(r+1) \geq d_i\right] &= \Pr\left[(w_i(r+1) - w_i(r)) \geq (d_i - w_i(r))\right] \\
&\geq \Pr\left[(w_i(r+1) - w_i(r)) \geq \frac{\Phi(r)}{2|U(r)|}\right] && \text{since } i \in L(r), \\
&\geq 1 - e^{-1/8} && \text{by the inequality above.}
\end{aligned}$$

The number of newly satisfied tasks in round r is at least $\sum_{i \in L(r)} \Pr[w_i(r+1) \geq d_i]$ in expectation. Therefore, the expected number of unsatisfied tasks at time $r+1$ is:

$$\begin{aligned}
\mathbb{E}\left[|U(r+1)|\right] &\leq |U(r)| - \sum_{i \in L(r)} \Pr[w_i(r+1) \geq d_i] \\
&\leq |U(r)| - \frac{|U(r)|}{2}\left(1 - e^{-1/8}\right) \\
&\leq |U(r)|\left(1 - \frac{1 - e^{-1/8}}{2}\right).
\end{aligned}$$

In this case, Equation (4.3) holds. □

Next, we consider the case of $c > 1$. Let d and k be arbitrary constants such that $0 < d < 1 - 1/c$ (so $c(1-d) > 1$) and $k = (1 - 1/c(1-d))(1 - e^{-cd^2/2})$.

Lemma 4.4.3. *For $c > 1$ and for each unsatisfied task $i \in U(r)$,* $\Pr[(w_i(r+1) - w_i(r)) < (1-d)c \cdot \Phi(r)/|U(r)|] \leq e^{-cd^2/2}$.

Proof. By Lemma 4.2.5, the number of inactive ants in round $r+1$ is at least $c \cdot \Phi(r)$. Therefore, for each $i \in U(r)$, $\mathbb{E}[w_i(r+1) - w_i(r)] \geq c \cdot \Phi(r)/|U(r)|$. By a Chernoff bound it follows that:

$$\begin{aligned}
&\Pr\left[(w_i(r+1) - w_i(r)) < \frac{(1-d)c\Phi(r)}{|U(r)|}\right] \\
&\leq \Pr\left[(w_i(r+1) - w_i(r)) < (1-d)\mathbb{E}\left[w_i(r+1) - w_i(r)\right]\right] \\
&\leq e^{-\frac{\mathbb{E}\left[w_i(r+1) - w_i(r)\right]d^2}{2}} \\
&\leq e^{-\frac{c\cdot\Phi(r)d^2}{2|U(r)|}} && \text{since } \Phi(r)/|U(r)| \geq 1, \\
&\leq e^{-\frac{cd^2}{2}}.
\end{aligned}$$

□

Unlike Theorem 4.4.5, where in round $r+1$ either the total deficit or the number of unsatisfied tasks decreases by a constant fraction, here we show that the number of unsatisfied tasks decreases by at least a constant fraction in round $r+1$ (this roughly corresponds to Case 2 in Theorem 4.4.5). We consider all tasks with a fairly low deficit, which are likely to get satisfied in a single round. The total deficit at time r is $\Phi(r)$, and the total number of inactive ants in round $r+1$ is at least $c \cdot \Phi(r)$. The fact that the number of inactive ants is at least a constant fraction greater than total deficit lets us show that the expected number of low-deficit tasks is at least a constant fraction of all unsatisfied tasks. Therefore, by satisfying these low-deficit tasks the number of unsatisfied tasks decreases by a constant fraction in expectation.

Lemma 4.4.4. *For $c > 1$,* $\mathbb{E}[|U(r+1)|] \leq |U(r)|(1-k)$.

Proof. Define task $i \in U(r)$ to have a *low deficit* if $\Phi_i(r) \leq (1-d)c\Phi(r)/|U(r)|$, and let $L(r) \subseteq U(r)$ denote the set of low-deficit tasks at time r. Similarly, let task $i \in U(r)$ have a *high deficit* if $\Phi_i(r) > (1-d)c\Phi(r)/|U(r)|$, and let $H(r) \subseteq U(r)$ denote the set of high-deficit tasks at time r. Therefore, $|U(r)| = |L(r)| + |H(r)|$.

Since the total deficit at time r is $\Phi(r)$, and each high-deficit task has deficit at least $(1-d)c\Phi(r)/|U(r)|$, it must be the case that:

$$|H(r)| \leq \frac{\Phi(r)}{(1-d)c\Phi(r)/|U(r)|} = \frac{|U(r)|}{c(1-d)}.$$

Therefore,

$$|L(r)| = |U(r)| - |H(r)| \geq |U(r)| \left(1 - \frac{1}{c(1-d)}\right).$$

For a task i with low deficit, the probability that it is satisfied at time $r+1$ is:

$$\begin{aligned}
\Pr\left[w_i(r+1) \geq d_i\right] &= \Pr\left[(w_i(r+1) - w_i(r)) \geq (d_i - w_i(r))\right] \\
&\geq \Pr\left[(w_i(r+1) - w_i(r)) \geq \frac{(1-d)c\Phi(r)}{|U(r)|}\right] \\
&\geq 1 - e^{-\frac{cd^2}{2}} \qquad \text{by Lemma 4.4.3.}
\end{aligned}$$

Therefore, the expected number of unsatisfied tasks at time $r+1$ is:

$$\begin{aligned}
\mathbb{E}\left[|U(r+1)|\right] &= |U(r)| - \sum_{i \in L(r)} \Pr\left[w_i(r+1) \geq d_i\right] \\
&\leq |U(r)| - |U(r)| \left(1 - \frac{1}{c(1-d)}\right)\left(1 - e^{-\frac{cd^2}{2}}\right) \\
&= |U(r)| \left(1 - \left(1 - \frac{1}{c(1-d)}\right)\left(1 - e^{-\frac{cd^2}{2}}\right)\right) \\
&= |U(r)| \left(1 - k\right).
\end{aligned}$$

□

Finally, we analyze the total running time of task allocation for an arbitrary probabilistic execution of the resulting system. Fix some arbitrary deterministic *success* components in each round. In the next theorem, we start at time 0, when the total deficit is $\Phi(0)$ and the number of unsatisfied tasks is at most $|T|$, and inductively apply Lemmas 4.4.2 and 4.4.4 and iterated expectation.

Theorem 4.4.5. *For $c \geq 1$ and for any δ, $0 < \delta < 1$, with probability at least $1 - \delta$, all tasks are satisfied by time* $\min\{|T|, 32(\ln \Phi(0) + \ln |T| + \ln(1/\delta))\}$.

Proof. By Corollary 4.2.2 and 4.2.4, both the number of unsatisfied tasks and the total deficit are monotonically non-increasing. Initially, the total deficit is $\Phi(0)$ and $|U(0)| \leq |T|$. Informally, by Lemma 4.4.2, in each round, either the number of unsatisfied tasks or the total deficit decreases by a constant fraction. So, if we consider r rounds, then either in at least $r/2$ rounds the total deficit decreases by a constant fraction, or in at least $r/2$ rounds the number of unsatisfied tasks decreases by a constant fraction. Formally, similarly to Theorem 4.3.4, we can inductively apply either Equation 4.2 or Equation 4.3 together with iterated expectation to show that for each $r \geq 0$, at least one of the following is true:

$$\mathbb{E}[|U(r)|] \leq |T| \left(1 - \frac{1 - e^{-1/8}}{2}\right)^{r/2}$$
$$\mathbb{E}[|\Phi(r)|] \leq \Phi(0) \left(1 - \frac{1}{16}\right)^{r/2}.$$

Therefore, for $r = 32(\ln \Phi(0) + \ln |T| + \ln(1/\delta))$, at least one of the following is true:

$$\mathbb{E}[|U(r)|] \leq |T| \left(1 - \frac{1 - e^{-1/8}}{2}\right)^{16(\ln \Phi(0) + \ln |T| + \ln(1/\delta))} \leq \delta$$
$$\mathbb{E}[|\Phi(r)|] \leq \Phi(0) \left(1 - \frac{1}{16}\right)^{16(\ln \Phi(0) + \ln |T| + \ln(1/\delta))} \leq \delta.$$

By a Markov bound, with probability at least $1 - \delta$, either $|U(r)| < 1$ or $\Phi(r) < 1$, implying that all tasks are satisfied by time r. Since *choice* always returns an unsatisfied task, by Lemma 4.2.6, all tasks are satisfied by time $|T|$. So, overall, with probability at least $1 - \delta$, all tasks are satisfied by time $\min\{|T|, 32(\ln \Phi(0) + \ln |T| + \ln(1/\delta))\}$. □

Corollary 4.4.6. *For $c \geq 1$ and for any δ and ϵ, $0 < \delta, \epsilon < 1$, with probability at least $1 - \delta$, the deficit at time* $\min\{|T|, 32(\ln |T| + \ln(1/\epsilon) + \ln(1/\delta))\}$ *is at most* $\epsilon \cdot \Phi(0)$.

Theorem 4.4.7. *For $c > 1$ and for any δ, $0 < \delta < 1$, with probability at least $1 - \delta$, all tasks are satisfied by time* $\min\{|T|, (2/\ln c)(\ln|T| + \ln(1/\delta))\}$.

Proof. Initially, $|U(0)| \leq |T|$. By Lemma 4.4.4, $\mathbb{E}[|U(r+1)|] \leq |U(r)|(1-k)$. Similarly to Theorem 4.3.4, we can inductively apply Lemma 4.4.4 and iterated expectation to show that for each $r \geq 0$, it is true that $\mathbb{E}[|U(r+1)|] \leq |T|(1-k)^r$. For $r = (\ln(1-k)^{-1})^{-1}(\ln|T| + \ln(1/\delta))$ we have:

$$\mathbb{E}[|U(r)|] \leq |T|(1-k)^{(\ln(1-k)^{-1})^{-1}(\ln|T|+\ln(1/\delta))} \leq \delta.$$

Therefore, by a Markov bound, with probability at least $1 - \delta$, $|U(r)| < 1$, implying that all tasks are satisfied by time $(\ln(1-k)^{-1})^{-1}(\ln|T| + \ln(1/\delta))$. Since *choice* always returns an unsatisfied task, by Lemma 4.2.6, all tasks are satisfied by time $|T|$, and the lemma follows. □

We can combine the results of Theorems 4.4.5 and 4.4.7. Clearly, if c is extremely close to 1, the $2/\ln c$ term becomes very large, and in the limit the running time becomes ∞. Therefore, we can take the minimum of the running times of Theorems 4.4.5 and 4.4.7 to get the overall running time of the algorithm.

Corollary 4.4.8. *For any δ, $0 < \delta < 1$, with probability at least $1 - \delta$, all tasks are satisfied by time* $\min\{|T|, \mathcal{O}(\ln^{-1} c) \cdot \mathcal{O}(\min\{|T|, \ln\Phi(0) + \ln|T| + \ln(1/\delta))\}\}$.

Corollary 4.4.9. *For any δ and ϵ, $0 < \delta, \epsilon < 1$, with probability at least $1 - \delta$, the deficit at time* $\min\{|T|, \mathcal{O}(\ln^{-1} c) \cdot \mathcal{O}(\ln(1/\epsilon) + \ln|T| + \ln(1/\delta))\}$ *is at most $\epsilon \cdot \Phi(0)$.*

4.5 Unsatisfied Tasks Prioritized by Deficit

In this section, we consider the third option for the *choice* component, defined in Section 4.1, where in each round *choice* returns a task $i \in U(r)$ with probability $(d_i - w_i(r))/\Phi(r)$. In Section 4.5.1, we analyze the time for ants to re-allocate when alternative tasks in each round are determined based on option (3). Then, in Section

4.5.2, we present an alternative model where each of the *success* and *choice* components provides noisy information to the ants. For the resulting noisy variant of option (3), we analyze the time for ants to re-allocate, satisfying the demands of the tasks approximately.

4.5.1 Option (3) with no Uncertainty

For Lemmas 4.5.1 and 4.5.2, assume α is a fixed execution and $r \geq 0$ is some fixed time in α. We consider the state variables at time r and the outputs of *success* in round $r+1$ to be fixed, and we consider the probability distribution over the randomness introduced by the *choice* outputs in round $r+1$.

Since an inactive ant starts working on a task i with probability $(d_i - w_i(r))/\Phi(r)$, and since there are at least $\Phi(r)$ inactive ants in round $r+1$, the expected number of new ants to join task i in round $r+1$ is at least a constant fraction of $d_i - w_i(r)$, which is exactly the deficit of the task at time r. In the next lemma, we show that each task is satisfied in round $r+1$ with a constant probability, and so the total number of unsatisfied tasks decreases by at least a constant fraction.

Lemma 4.5.1. *For $c \geq 1$, $\mathbb{E}[|U(r+1)|] \leq |U(r)|/2$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(r)/2$.*

Proof. We start by bounding the probability $\Pr[w_i(r+1) \geq d_i]$ from below, for some task $i \in U(r)$. We can express $(w_i(r+1) - w_i(r))$ as a binomial variable that is the sum of n independent identical random variables, each with probability p of success. By Lemma 4.2.5, the number of inactive ants in round $r+1$ is at least $\Phi(r)$, so $n \geq \Phi(r)$, and by the definition of *choice*, we know that $p = (d_i - w_i(r))/\Phi(r)$. By [72], the median m of $(w_i(r+1) - w_i(r))$ is:

$$\left\lfloor \frac{n(d_i - w_i(r))}{\Phi(r)} \right\rfloor \leq m \leq \left\lceil \frac{n(d_i - w_i(r))}{\Phi(r)} \right\rceil.$$

Since we want to lower-bound $\Pr[w_i(r+1) - w_i(r) \geq d_i - w_i(r)]$, we can consider $n = \Phi(r)$ because the probability that task i is satisfied only increases if we increase n. Therefore, $m = d_i - w_i(r)$, and by the definition of the median it follows that $\Pr[w_i(r+1) - w_i(r) \geq d_i - w_i(r)] \geq 1/2$.

By Lemma 4.2.7, $\mathbb{E}[|U(r+1)|] \leq |U(r)|/2$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(r)/2$. □

Next, we consider the case of $c > 1$. Similarly to Section 4.4, we show that each task is satisfied with a constant probability, so the number of unsatisfied tasks and the total deficit decrease by a constant fraction in each round.

Lemma 4.5.2. *For $c > 1$, $\mathbb{E}[|U(r+1)|] \leq |U(r)| \cdot e^{-c(1-1/c)^2/2}$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(r) \cdot e^{-c(1-1/c)^2/2}$.*

Proof. By Lemma 4.2.5, the number of inactive ants in round $r+1$ is at least $c \cdot \Phi(r)$. Therefore, for each $i \in U(r)$, $\mathbb{E}[w_i(r+1) - w_i(r)] \geq c \cdot \Phi(r)(d_i - w_i(r))/\Phi(r) = c \cdot (d_i - w_i(r))$. By a Chernoff bound it follows that:

$$
\begin{aligned}
&\Pr\left[(w_i(r+1) - w_i(r)) < (d_i - w_i(r))\right] \\
&\leq \Pr\left[(w_i(r+1) - w_i(r)) < \left(\frac{1}{c}\right)\mathbb{E}\left[w_i(r+1) - w_i(r)\right]\right] \\
&\leq e^{-\frac{\mathbb{E}[w_i(r+1)-w_i(r)]\left(1-\frac{1}{c}\right)^2}{2}} \\
&\leq e^{-\frac{c\cdot(d_i-w_i(r))\left(1-\frac{1}{c}\right)^2}{2|U(r)|}} && \text{since } (d_i - w_i(r)) \geq 1, \\
&\leq e^{-\frac{c\left(1-\frac{1}{c}\right)^2}{2}}.
\end{aligned}
$$

By Lemma 4.2.7, $\mathbb{E}[|U(r+1)|] \leq |U(r)| \cdot e^{-c(1-1/c)^2/2}$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(r) \cdot e^{-c(1-1/c)^2/2}$. □

Finally, we fix some arbitrary deterministic *success* components in each round and we analyze the total running time of task allocation for an arbitrary probabilistic execution of the resulting system. In the next theorem, we start at time 0, when the number of unsatisfied tasks is at most $|T|$, and inductively apply Lemmas 4.5.1 and 4.5.2 and iterated expectation.

Theorem 4.5.3. *For $c \geq 1$ and for any δ, $0 < \delta < 1$, with probability at least $1 - \delta$, all tasks are satisfied by time $\min\{|T|, \min\{\log|T|, \log\Phi(0)\} + \log(1/\delta)\}$.*

Proof. Initially, $|U(0)| \leq |T|$. By Lemma 4.5.1, $\mathbb{E}[|U(r+1)|] \leq |U(r)|/2$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(r)/2$. Similarly to Theorem 4.3.4, we can inductively apply Lemma 4.5.1

and iterated expectation to show that for each $r \geq 0$, it is true that $\mathbb{E}[|U(r+1)|] \leq |T|(1/2)^r$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(0)(1/2)^r$. For $r = \min\{\log|T| + \log\Phi(0)\} + \log(1/\delta)$ we have that at least one of the following is true:

$$\mathbb{E}[|U(r)|] \leq |T| \cdot 2^{-(\min\{\log|T|, \Phi(0)\} + \log(1/\delta))} \leq \delta$$
$$\mathbb{E}[\Phi(r)] \leq \Phi(0) \cdot 2^{-(\min\{\log|T|, \Phi(0)\} + \log(1/\delta))} \leq \delta.$$

Therefore, by a Markov bound, with probability at least $1-\delta$, either $|U(r)| < 1$ or $\Phi(0) < 1$, implying that all tasks are satisfied by time $\min\{\log|T|, \log\Phi(0)\} + \log(1/\delta)$. Since *choice* always returns an unsatisfied task, by Lemma 4.2.6, all tasks are satisfied by time $|T|$, and the lemma follows. □

Corollary 4.5.4. *For $c \geq 1$ and for any δ and ϵ, $0 < \delta, \epsilon < 1$, with probability at least $1-\delta$, the deficit at time $\min\{|T|, \log(1/\epsilon) + \log(1/\delta)\}$ is at most $\epsilon \cdot \Phi(0)$.*

Theorem 4.5.5. *For $c > 1$ and for any δ, $0 < \delta < 1$, with probability at least $1-\delta$, all tasks are satisfied by time $\min\{|T|, (2/(c(1-1/c)^2))(\min\{\ln|T|, \ln\Phi(0)\} + \ln(1/\delta))\}$.*

Proof. Initially, $|U(0)| \leq |T|$. By Lemma 4.5.2, $\mathbb{E}[|U(r+1)|] \leq |U(r)| \cdot e^{-\frac{c\left(1-\frac{1}{c}\right)^2}{2}}$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(r) \cdot e^{-\frac{c\left(1-\frac{1}{c}\right)^2}{2}}$. Similarly to Theorem 4.3.4, we can inductively apply Lemma 4.5.1 and iterated expectation to show that for each $r \geq 0$, it is true that $\mathbb{E}[|U(r+1)|] \leq |T|(e^{-rc(1-1/c)^2/2})$ and $\mathbb{E}[\Phi(r+1)] \leq \Phi(0)(e^{-rc(1-1/c)^2/2})$. For $r = (2/(c(1-1/c)^2))(\min\{\ln|T|, \ln\Phi(0)\} + \ln(1/\delta))$ we have that at leas one of the following is true:

$$\mathbb{E}[|U(r)|] \leq |T| \cdot e^{-(2/(c(1-1/c)^2))(\min\{\ln|T|, \ln\Phi(0)\} + \ln(1/\delta))} \leq \delta$$
$$\mathbb{E}[\Phi(r)] \leq \Phi(0) \cdot e^{-(2/(c(1-1/c)^2))(\min\{\ln|T|, \ln\Phi(0)\} + \ln(1/\delta))} \leq \delta.$$

Therefore, by a Markov bound, with probability at least $1-\delta$, either $|U(r)| < 1$ or $\Phi(r) < 1$, implying that by time $(2/(c(1-1/c)^2))(\min\{\ln|T|, \ln\Phi(0)\} + \ln(1/\delta))$ all tasks are satisfied. Since *choice* always returns an unsatisfied task, by Lemma 4.2.6, all tasks are satisfied by time $|T|$, and the lemma follows. □

Corollary 4.5.6. *For $c > 1$ and for any δ and ϵ, $0 < \delta, \epsilon < 1$, with probability at least $1 - \delta$, the deficit at time $\min\{|T|, (2/(c(1-1/c)^2))(\log(1/\epsilon) + \log(1/\delta))\}$ is at most $\epsilon \cdot \Phi(0)$.*

We can combine the results of Theorems 4.5.3 and 4.5.5. Clearly, if c is extremely close to 1, the $1/(c(1-1/c)^2)$ term becomes very large, and in the limit the running time becomes ∞. Therefore, we can take the minimum of the running times of Theorems 4.5.3 and 4.5.5 to get the overall running time of the algorithm.

Corollary 4.5.7. *For any δ, $0 < \delta < 1$, with probability at least $1 - \delta$, all tasks are satisfied by time $\min\{|T|, \mathcal{O}(c^{-1}) \cdot \mathcal{O}(\log \Phi(0) + \log(1/\delta))\}$.*

For $c = 2 + \sqrt{3} \approx 3.7$, we have $2/(c(1-1/c)^2) = 1$, so we can think of the running time being bounded by Theorem 4.5.3 for $c \in [1, 2 + \sqrt{3}]$ and bounded by Theorem 4.5.5 for $c > 2 + \sqrt{3}$.

Corollary 4.5.8. *For any δ and ϵ, $0 < \delta, \epsilon < 1$, with probability at least $1 - \delta$, the deficit at time $\min\{|T|, \mathcal{O}(c^{-1}) \cdot \mathcal{O}(\log(1/\epsilon) + \log(1/\delta))\}$ is at most $\epsilon \cdot \Phi(0)$.*

4.5.2 Option (3) under Uncertainty

Suppose the *success* component is not completely reliable and it can flip the 0/1 bits of at most $0 \leq z \leq |A|$ ants in round $r + 1$. Moreover, we assume the information needed to determine the outputs of the *choice* component in the same round is based on the state variables at time r. That is, the outputs of *choice* in round $r + 1$ do not incorporate the outputs of *success* (with the z potential mistakes) in round $r + 1$.

Also, suppose the *choice* component is also not completely reliable and can change the probability of outputting task i from exactly $\Phi_i(r)/\Phi(r)$ to any value larger than $(1-y)(\Phi_i(r)/\Phi(r))$ for any $0 \leq y < 1$ while still maintaining a probability distribution over all the tasks.

In other words, we have an alternative model where we consider two types of uncertainty: the number of successful ants and the probabilities with which tasks are assigned to ants are not exact but bounded.

Next, we analyze the time for ants to re-allocate in this modified model. The statements and proofs below are very similar to the ones in Section 4.5.1 with the following two main differences. First, it is no longer possible to guarantee that all tasks are satisfied but we can show that the deficit does not exceed z. Second, whenever the *choice* component is supposed to return a given task with some probability p, we use the lower bound $p(1-y)$ for that probability; this results in a running time that increases as y approaches 1.

For Lemmas 4.5.9 and 4.5.10, assume α is a fixed execution and $r \geq 0$ is some fixed time in α. We consider the state variables at time r and the outputs of *success* in round $r+1$ to be fixed, and we consider the probability distribution over the randomness introduced by the *choice* outputs in round $r+1$.

For each task $i \in T$, let z_i^0 be the number of 0's flipped to 1's, and let z_i^1 be the number of 1's flipped to 0's by *success* in round $r+1$. Let $z^0 = \sum_{i \in T} z_i^0$ and $z^1 = \sum_{i \in T} z_i^1$, so $z^0 + z^1 \leq z$.

Note that, based on the definitions above, the number of workers $w_i(r)$ working at task i decreases by z_i^1 before the unsuccessful ants choose a new task to join. Also, the number of inactive ants is at least $\Phi(r) - z_i^0$ because z_i^0 ants are informed they are successful while they are actually not successful.

Lemma 4.5.9. *For $c \geq 1$, $\mathbb{E}[\Phi(r+1)] \leq (1/4)((3+y)\Phi(r) + z)$.*

Proof. Let random variable $X_i(r+1)$ denote the number of ants that join task i in round $r+1$. The probability for an ants to receive task i from *choice* in round $r+1$ is $p_i \in [(1-y)(\Phi_i(r)/\Phi(r)), (1+y)(\Phi_i(r)/\Phi(r))]$. By Lemma 4.2.5, the number of inactive ants in round $r+1$ is at least $\Phi(r)$; however, now they may be fewer if *success* flipped some 0's to 1's, so the number of inactive ants is at least $\Phi(r) - z^0$.

So, $\mathbb{E}[X_i(r+1)] \geq p_i \cdot (\Phi(r) - z^0)$ and the expected value of $\Phi(r+1)$ is:

$$\begin{aligned}
\mathbb{E}[\Phi(r) - \Phi(r+1)] &= \sum_{i \in T} \mathbb{E}[\min\{X_i(r+1) - z_i^1, (d_i - w_i(r))\}] \\
&\geq \sum_{i \in T} \left(\frac{1}{4}\right) \min\{\mathbb{E}[X_i(r+1)] - z_i^1, (d_i - w_i(r))\} \\
&\geq \sum_{i \in T} \left(\frac{1}{4}\right) \min\left\{p_i \cdot (\Phi(r) - z^0) - z_i^1, \Phi_i(r)\right\} \\
&\geq \left(\frac{1}{4}\right) \sum_{i \in T} (1-y) \cdot \Phi_i(r) - p_i \cdot z^0 - z_i^1 \\
&\geq \left(\frac{1}{4}\right) \left((1-y) \cdot \Phi(r) - z^0 - z^1\right) \\
&\geq \left(\frac{1}{4}\right) \left((1-y) \cdot \Phi(r) - z\right).
\end{aligned}$$

□

Lemma 4.5.10. *For $c > 1$, $\mathbb{E}[\Phi(r) - \Phi(r+1)] \geq (1 - e^{-(c(1-1/c)^2)/2})((1-y) \cdot \Phi(r) - z)$.*

Proof. Let that random variable $X_i(r+1)$ denote the number of ants that join task i in round $r+1$. We assumed that the outputs of *choice* are based only on $w_i(r)$, so the probability for an ants to receive task i from *choice* in round $r+1$ is $p_i \in [(1-y)(\Phi_i(r)/\Phi(r)), (1+y)(\Phi_i(r)/\Phi(r))]$ regardless of the outputs of *success* in round $r+1$. By Lemma 4.2.5, the number of inactive ants in round $r+1$ is at least $c \cdot \Phi(r)$. However, after the mistakes of the *success* component, the number of inactive ants is at least $c \cdot \Phi(r) - z^0 \geq c \cdot (\Phi(r) - z^0)$. So, $\mathbb{E}[X_i(r+1)] \geq p_i c \cdot (\Phi(r) - z^0)$.

By a Chernoff bound, it follows that:

$$\Pr\left[X_i(r+1) < p_i(\Phi(r) - z^0)\right] \leq \Pr\left[X_i(r+1) < \left(\frac{1}{c}\right) \mathbb{E}[X_i(r+1)]\right] < e^{-\frac{c(1-\frac{1}{c})^2}{2}}.$$

By linearity of expectation it follows that:

$$\begin{aligned}
\mathbb{E}[\Phi(r) - \Phi(r+1)] &= \sum_{i \in T} \mathbb{E}[\Phi_i(r) - \Phi_i(r+1)] \\
&\geq \sum_{i \in T} \mathbb{E}[\Phi_i(r) - \Phi_i(r+1) \mid X_i(r+1) > p_i(\Phi(r) - z^0)] \\
&\qquad \cdot \Pr[X_i(r+1) > p_i(\Phi(r) - z^0)] \\
&\geq \sum_{i \in T} \min\{p_i(\Phi(r) - z^0) - z_i^1, \Phi_i(r)\} \\
&\qquad \cdot \Pr[X_i(r+1) > p_i(\Phi(r) - z^0)] \\
&\geq \sum_{i \in T} \left(p_i(\Phi(r) - z^0) - z_i^1\right) \cdot \left(1 - e^{-\frac{c(1-\frac{1}{c})^2}{2}}\right) \\
&\geq \left(1 - e^{-\frac{c(1-\frac{1}{c})^2}{2}}\right) \sum_{i \in T} (1-y)\Phi_i(r) - p_i z^0 - z_i^1 \\
&\geq \left(1 - e^{-\frac{c(1-\frac{1}{c})^2}{2}}\right) ((1-y)\Phi(r) - z).
\end{aligned}$$

□

Finally, we fix some arbitrary deterministic *success* components in each round and we analyze the total running time of task allocation for an arbitrary probabilistic execution of the resulting system. In the next theorem, we start at time 0 and inductively apply Lemmas 4.5.9 and 4.5.10 and iterated expectation.

Theorem 4.5.11. *For $c \geq 1$, for any δ, $0 < \delta < 1$, and for $r = (1/\ln(4/(3+y)))(\ln \Phi(0) + \ln(1/\delta))$, $\Pr[\Phi(r) \leq z] \geq 1 - \delta$.*

Proof. By Lemma 4.5.9, $\mathbb{E}[\Phi(r+1)] \leq (1/4)((3+y)\Phi(r) + z)$. Similarly to Theorem 4.3.4, we can inductively apply Lemma 4.5.9 and iterated expectation to show that for each $r \geq 0$, it is true that:

$$\mathbb{E}[\Phi(r+1)] \leq \Phi(0)\left(\frac{3+y}{4}\right)^r + z\left(\frac{(4/(1-y))^r - 1}{(4/(1-y))^r}\right) \leq \Phi(0)\left(\frac{3+y}{4}\right)^r + z.$$

For $r = (1/\ln(4/(3+y)))(\ln\Phi(0) + \ln(1/\delta))$ we have that:

$$\mathbb{E}[\Phi(r) - z] \leq \Phi(0) \cdot \left(\frac{3+y}{4}\right)^{(1/\ln(4/(3+y)))(\ln\Phi(0)+\ln(1/\delta))} \leq \delta.$$

Therefore, by a Markov bound, $\Pr[\Phi(r) - z \geq 1] \leq \delta$, so $\Pr[\Phi(r) - z < 1] \geq 1 - \delta$, implying that with probability at least $1 - \delta$, $\Phi(r) - z \leq 0$, and so $\Pr[\Phi(r) \leq z] \geq 1 - \delta$. □

Let $k = (1-y)(1 - e^{-(c(1-1/c)^2)/2})$.

Theorem 4.5.12. *For $c > 1$, for any δ, $0 < \delta < 1$, and for round $r = (\ln(1-k)^{-1})^{-1}(\ln\Phi(0) + \ln(1/\delta))$, $\Pr[\Phi(r) \leq z] \geq 1 - \delta$.*

Proof. Similarly to Theorem 4.5.11, we can inductively apply Lemma 4.5.10 and iterated expectation to show that for each $r \geq 0$, it is true that:

$$\mathbb{E}[\Phi(r+1)] \leq \Phi(0)\left(1 - (1-y)\left(1 - e^{-\frac{c(1-\frac{1}{c})^2}{2}}\right)\right)^r + z = \Phi(0) \cdot (1-k)^r + z.$$

For $r = (\ln(1-k)^{-1})^{-1}(\ln\Phi(0) + \ln(1/\delta))$ we have that:

$$\mathbb{E}[\Phi(r) - z] \leq \Phi(0) \cdot (1-k)^r \leq \Phi(0) \cdot e^{-(\ln\Phi(0)+\ln(1/\delta))} \leq \delta.$$

Therefore, by a Markov bound, $\Pr[\Phi(r) - z \geq 1] \leq \delta$, so $\Pr[\Phi(r) - z < 1] \geq 1 - \delta$, implying that with probability at least $1 - \delta$, $\Phi(r) - z \leq 0$, and so $\Pr[\Phi(r) \leq z] \geq 1 - \delta$. □

Corollary 4.5.13. *For any δ, $0 < \delta < 1$, and for $r = \min\{1/\ln(4/(3+y)), (\ln(1-k)^{-1})^{-1}\}(\ln\Phi(0) + \ln(1/\delta)) = \mathcal{O}(\max\{c^{-1}, \ln^{-1}(y^{-1})\})(\ln\Phi(0) + \ln(1/\delta))$, $\Pr[\Phi(r) \leq z] \geq 1 - \delta$.*

4.6 Discussion

4.6.1 Summary of Results

Table 1-1 (reproduced below) summarizes our results. One of the main goals of these results is to highlight the dependence of the time for ants to re-allocate on various colony and environment parameters. In all three options for the *choice* component, we see that the time for ants to re-allocate is logarithmic in the amount of work Φ and does not directly depend on the colony size $|A|$. Furthermore, in each of those three cases, the time for ants to re-allocate decreases as the ants-to-work ratio, c, increases. This relationship is not exactly the same in all three cases: in options (1) and (3) the dependence is inversely proportional and linear in c, while in option (2) it is inversely proportional and logarithmic in c. Figure 4-2 summarizes and illustrates our results in the three options for the *choice* component with respect to c. Note that due to the weaker dependence of option (2) on the value of c, initially it performs better than option (1), but as c increases, option (1) corresponds to better running times.

	option (1)	option (2)	option (3)
satisfy all Φ work with prob. $\geq 1-\delta$	$\mathcal{O}(\lvert T\rvert(\frac{1}{c}))$ $(\ln\Phi + \ln(\frac{1}{\delta}))$	$\min\{\lvert T\rvert,$ $(\min\{1, \mathcal{O}(\frac{1}{\ln c})\}\cdot$ $(\ln\Phi + \ln(\frac{1}{\delta})))\}$	$\min\{\lvert T\rvert, \mathcal{O}(\frac{1}{c})$ $(\ln\Phi + \ln(\frac{1}{\delta}))\}$
satisfy $\Phi(1-\epsilon)$ work with prob. $\geq 1-\delta$	$\mathcal{O}(\lvert T\rvert(\frac{1}{c}))$ $(\ln(\frac{1}{\epsilon}) + \ln(\frac{1}{\delta}))$	$\min\{\lvert T\rvert,$ $(\min\{1, \mathcal{O}(\frac{1}{\ln c})\}\cdot$ $(\ln(\frac{1}{\epsilon}) + \ln(\frac{1}{\delta})))\}$	$\min\{\lvert T\rvert, \mathcal{O}(\frac{1}{c})$ $(\ln(\frac{1}{\epsilon}) + \ln(\frac{1}{\delta}))\}$
satisfy $\Phi - z$ work under uncertainty	did not analyze	did not analyze	$\min\{\lvert T\rvert, (\ln\Phi + \ln(\frac{1}{\delta}))$ $\mathcal{O}(\max\{\frac{1}{c}, \frac{1}{\ln(1/y)}\})\}$

Naturally, all three of these results also depend on the probability $1-\delta$ with which we require the tasks to be satisfied. In the results where work needs to be satisfied only approximately (the second row of the table), we no longer see a dependence on Φ; in this case, the significant parameter that affects the running time is ϵ – the fraction of work that may be unsatisfied at the end of the task re-allocation.

Additionally, in option (1), we see a linear dependence on the number $|T|$ of tasks due to the slow sampling of unsatisfied tasks in this case. In options (2) and (3),

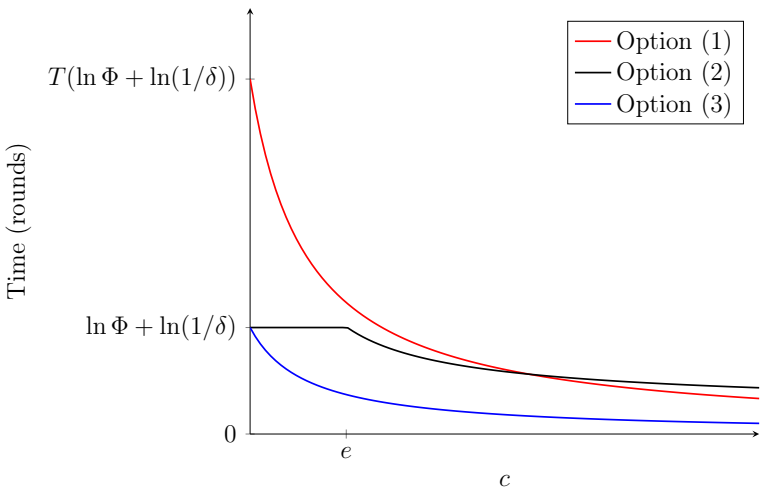

Figure 4-2: The three plots indicate the times until workers re-allocate successfully for options (1), (2), and (3) of the *choice* component as a function of c. For options (1) and (3) the plotted function is approximately $1/c$, and for option (2), the plotted function is approximately $\min\{1, 1/\ln c\}$. We multiply these functions by the corresponding time to re-allocate for $c = 1$. For options (2) and (3), the running times technically also depend on $|T|$, but for simplicity we do not depict the min function.

we have a minimum over $|T|$ and the remaining expressions indicating that when the number of tasks is very small, the other parameters (the total amount of work Φ, the probability $1 - \delta$ of satisfying the tasks, and the fraction $1 - \epsilon$ of work to be satisfied) are irrelevant for the efficiency of task allocation.

Since task allocation is the fastest under option (3), we also analyze the performance of the algorithm under uncertainty. We assume an adversary can arbitrarily flip at most z bits of the *success* component outputs, and the probabilities with which the *choice* component outputs tasks is lower-bounded by a $(1 - y)$ factor of the original probabilities. In order to satisfy at least $\Phi - z$ units of work, ants need to pay an extra multiplicative factor of at most $1/\ln(1/y)$.

4.6.2 Biological Implications

Modeling, in general, can serve different purposes in the scientific process [63, 109]. From a biological viewpoint, our goal is to examine whether task allocation is a difficult problem, and what factors affect the various task allocation strategies.

4.6.2.1 Is task allocation a difficult problem in biological systems?

From a biological perspective, a problem is considered difficult if it requires a significant amount of some resource (for example, energy or time) to solve the problem. If task allocation is an easy problem, then the match of work to workers can be achieved without significant costs in terms of the resource of choice (in our setting, time). In complex systems where task allocation is difficult, on the other hand, the choice of task allocation algorithm is crucial for system performance; in biological systems where this is the case, we would expect task allocation mechanisms to be under strong (evolutionary) selection, and their evolution to reflect the specific ecological context of the system. In social insect colonies, for example, task allocation mechanisms appear to differ between species – this could be the case because different species have developed different, equally good, solutions, or because different species have different requirements (because they differ in the frequency with which demand for work in different tasks changes). There is some evidence that even brief mismatches of work to workers (incorrect task allocation) can be detrimental in certain species, for example, because brood do not develop well when briefly not thermoregulated[1] [62]. This implies that certain species are likely to use fast task allocation strategies like options (2) and (3) rather than slower ones like option (1).

In order to gain insight into the difficulty of the task allocation problem, we estimate the time to correct allocation for several species and contexts (Figure 4-3) by substituting specific values into the running time expressions we derived in Sections 4.3 – 4.5. Some of the values of the parameters that we use correspond to empirical observations from real experiments (see Figure 4-4 for a more information on the values and the corresponding experiments).

For example, we estimate that when a honey bee colony is attacked by a large predator, and 5000 ($\pm 30\%$) bees should ideally be allocated to defense, the time to achieve this within our generalized task allocation algorithm would be around $5 - 10$ rounds if all bees can directly sense the need for more defenders (options (2) or (3)),

[1] Brood of some species requires a specific temperature to be constantly maintained in the nest. Bees, for example, can regulate the nest temperature by flapping their wings.

Insect name	$\|T\|$	c	Φ	$1-\delta$	$1-\epsilon$	(1)	(2)	(3)
Honey bee (Apis mellifera) predator attack	10	1.3	5000	0.95	0.7	708.5 (258.4)	10 (10)	6.3 (4.7)
Honey bee (Apis mellifera) change in foraging conditions	10	1.3	150	0.8	0.7	407.3 (173.1)	10 (10)	4.9 (3.4)
Rock ants (Temnothorax rugutulus) change in foraging conditions	4	1.7	5	0.5	0.7	43.3 (35.7)	4 (4)	2.7 (2.4)
Rock ants (Temnothorax rugutulus) emigration after nest breakdown	4	1.7	25	0.9	0.9	103.9 (86.7)	4 (4)	4 (4)
Bumble bee (Bombus impatiens)	8	1.5	5	0.9	0.75	166.9 (157.4)	8 (8)	4.6 (4.3)

Figure 4-3: Numerical Results. For each option, we calculate the number of rounds until the entire initial deficit Φ is satisfied and, in parentheses, the number of rounds until a $(1-\epsilon)\cdot\Phi$ fraction of the deficit is satisfied. These are not intended to be exact time estimates; the values for c, δ, and ϵ have not been estimated empirically for any species, nor is it clear how long a round precisely should be. The intent, here, is to check whether task allocation might take a significant amount of time in realistic scenarios. These numerical estimates also serve to illustrate how the different parameters affect the time to successful reallocation in a realistic context of other parameter values.

and 700 rounds if they cannot (and only arrive in the defense task because they randomly tested different tasks in different rounds, option (1)). Since this particular situation requires a quick collective response, the difference between option (1) and options (2) or (3) appears significant, regardless of whether a round takes minutes or seconds to complete.

In another example, a change in foraging conditions in the case of rock ants (*Temnothorax*) may imply that only five additional workers need to be allocated to the task of foraging; however, in that system it appears likely that individuals need on the order of a minute rather than seconds to assess both the state of their environment and whether their own task performance is successful. If that is the case, a delay of

40 rounds may also be a significant and costly delay to appropriately exploit novel food sources, for example.

In all cases, the crucial factors in the task allocation process are whether or not individuals can assess the demand across different tasks simultaneously (instead of only in the one task they are working on), and what time period a round corresponds to (i.e. how long it takes a worker to assess whether its current work is needed). Overall, our calculations show that realistic parameter estimates can lead to potentially significant costs of slow task allocation. Our calculations are coarse since the precise values of many of the parameters are not known (however, see Figure 4-4 for references on parameter estimates). More empirical work in this area would be useful.

4.6.2.2 Colony size does not directly affect the efficiency of task allocation

Contrary perhaps to conventional wisdom in both biology and computer science, we do not find a direct dependence of the time to solve the task allocation problem on the colony size. This holds even if all the work has to be satisfied only with a certain probability, and only close to the total needed work. This result is perhaps expected because we modeled neither the type of noise that would lead to a benefit of large numbers (where the relative amount of variation in environments decreases with colony size), nor did we implement any economies of scale (no broadcast signals or any other communication mechanisms). Although this reasoning is sensible in hindsight, it was not what we had initially expected nor what is suggested in the literature [44].

4.6.2.3 The workers-to-work ratio affects the efficiency of task allocation

We discover that to understand the dependence of task allocation on the number of workers in the colony, actually what we really need to know is the total amount of work that needs to be done. This total amount of work available (or necessary) has not been studied explicitly either empirically or in models of social insect task allocation, with a few exceptions [41]. So, we do not have a good understanding of how the total amount of work behaves with respect to the colony size intra- or inter-specifically.

Here we have simply assumed that the ratio of the colony size and the total amount of work is constant, but this may well not generally be the case. Previous studies and conceptual papers have suggested either that larger colonies are relatively less productive, perhaps suggesting that less work is available per worker, or that they are more productive because they are capitalizing on some economies of scale; it is unclear what the latter would imply for the amount of work per worker available. One interesting new hypothesis here is that the evolution of task allocation across social insects may, in part, be driven by the factors that limit productivity (for example, the colony raising brood at near the queen's maximal egg laying rate). In this case the total amount of work may increase less than linearly with increasing colony size, and thus task allocation may become easier, even trivial, at higher colony sizes. Our modeling study thus suggests a new hypothesis (one for the purposes of modeling more generally, [55]), by providing the insight that a previously ignored parameter (the workers-to-work ratio c) impacts the outcome of a well-studied process.

4.6.2.4 Extra workers make task allocation faster

Inactive workers are common in social insect colonies. Possible reasons for this inactivity include selfish workers [26, 69], immature workers [24], or temporarily unemployed workers due to fluctuating total demand [25]. We have shown that the workers-to-work ratio c generally leads to faster task allocation. This is a novel hypothesis for the existence of inactive workers in social insect colonies and other complex systems [25]. That is, colonies may produce more workers than needed to complete available work simply in order to speed up the process of (re-)allocating workers to work, and thus potentially reducing costs of temporary mismatches of workers with needed work. In other words, inactive (surplus) workers in colonies may increase colony flexibility and task allocation efficiency in environments where task demands often change and workers frequently have to be reallocated. The benefit of extra workers does not depend on colony size, thus we would expect both large and small colonies to have as many extra workers as they can afford. Although the dependence on c varies with task allocation algorithm, higher c is always beneficial.

4.7 Open Problems

We have explored a range of models that vary in the feedback ants receive from the environment in each step. Since the goal of this work is to match real insect behavior, although abstractly, it would be interesting to design other models of interaction between ants and tasks (through the environment or not) that match specific ant species behavior.

In particular, the *threshold-based model* [15] is a widely-accepted model of task allocation among biologists. In the threshold-based model, each ant has a value (or a set of values) that represents its preference/willingness to start work on a given task. As the ant encounters different tasks, it uses this value as a threshold to determine what task to work on. This model has many variations in terms of how many thresholds each ant has, whether they change with time or not, whether the changes are short-lived or long-lived, what distribution the thresholds come from.

A specific problem related to the threshold-based model is to determine what is the optimal distribution from which to choose the thresholds of each ant for each task such that the ants will be able to satisfy the most amount of work. Note that if all ants are fairly unwilling to start working, then many tasks will remain unsatisfied. An even more specific question is to determine how efficient task allocation is under simple known distributions; for example, each threshold is chosen uniformly at random from some range.

Threshold-based models introduce some differences among the ant workers in terms of their task preferences. However, an even more realistic extensions is to assume different ants can contribute different amounts of work/energy to different tasks. As noted in [28], this problem is NP-hard in its most general form. A challenging open problem is to develop heuristics and approximations to the general solution; for example, ants may not be able to satisfy the demands of all the tasks, but they can satisfy a large fraction of them. One approach is to represent the tasks and their demands as a linear program and then use the distributed multiplicative weights update method for solving linear programs [43].

Finally, so far we have abstracted away from the process through which ants discover tasks and their demands. In real ant colonies, tasks differ in the method through which an ant can sense task demands. For example, if the nest is overheating, all ants in the nest can instantaneously sense the need for cooling down. However, if some larvae in some specific nest location are underfed, it may take an ant some time to discover the need to feed them. A real-world model of task allocation would include different subroutines for ants to search for tasks and evaluate their demands. These subroutines can also include a spatial component where ants walk randomly (or using some other rule) in the nest discovering and evaluating tasks.

Symbol	Parameter definition	Plausible range	Explanation for range	References
$\|T\|$	number of tasks	$[2, 20]$	At low end if conceived of as the number of distinct worker task groups; at higher end if all identifiable worker activities are included.	[22, 111, 66, 112]
Φ	initial deficit	$[5, 500]$	Considerable variation across species and situations; what is empirically measured is the number of workers actually re-allocated or activated.	[106, 37, 42, 40, 70]
$\|A\|$	number of workers	[2, 20 million]	Most species are in the $10 - 500$ range for total colony size.	[44]
c	fraction of extra workers	$[1, 2]$	Since the total amount of work has not been empirically measured, neither has c. If we assume inactive workers may be in excess of work that needs to be performed, values in the entire range are plausible.	[22, 66, 71, 98, 105]
$1 - \delta$	success probability	$[0.5, 0.95]$	To our knowledge, no attempts to estimate δ exist. Our estimates for Figure 4-3 are simply based on the assumption that in some cases, e.g. defense, colonies would need to be very certain that approximately the correct number of workers are allocated to the task at hand; in other cases, such as foraging, colonies may only need moderate certainty that task allocation is successful.	
$1 - \epsilon$	fraction of deficit to be satisfied	$[0.7, 0.9]$	ϵ reflects the degree to which the demand for work in a task is exactly matched. Given the high degree of stochasticity observed in task allocation in social insects, we assumed here that $1 - \epsilon$ is not required to be very close to 1 in most cases.	[37, 23]

Figure 4-4: Summary of parameters used in the task allocation model and analysis.

Chapter 5

Contributions and Significance

In this chapter, we summarize the main implications of the results in the previous chapters, both with respect to contributions to theoretical distributed computing and evolutionary biology. The rest of this chapter is structured around the two main goals of biological distributed algorithms: (1) use tools and techniques from distributed computing to gain insight into real biological behavior, and (2) learn from the models and algorithms occurring in ant colonies with the goal of designing better distributed algorithms.

Our results in foraging and house hunting generally refer to the second goal above, and their significance can best be described in terms of the lessons we have learned as theoretical computer scientists from the structure and behavior of insect colonies. These lessons include focusing on new more meaningful metrics besides the standard time and message complexity, looking to biology for natural lower bounds that do not exploit the weaknesses of the models to an extreme, and striving for simple algorithms with new robustness properties. Our results in task allocation refer to the first goal above, and can best be attributed to providing biologists with a new direction for hypothesis generation about insect behavior.

5.1 Lessons for Theoretical Distributed Computing Scientists

The standard approach to problems in distributed computing usually has the following structure: given a fixed mathematical model of computation, we define a problem, design algorithms that correctly solve the problem, analyze the performance of the algorithms with respect to time, space and message complexity metrics, and prove lower bounds on the minimum amount of resources (again, in terms of the same metrics) needed to solve the problem in the given model. We suggest a few changes in this structure, supported by evidence from our results on foraging and house hunting. The goals are to have results more widely applicable to real systems, more relevant to biological systems, and possibly more interesting and meaningful from a theoretical viewpoint.

5.1.1 New Metrics

As computer scientists, we are used to analyzing the efficiency of algorithms in terms of time, space, and message complexity. In our foraging work, however, we defined a new combined metric and showed that it captures more comprehensively the nature of the search problem, compared to any other known single metric. What advantage does a combined metric give us over simply considering different metrics in isolation and proving trade-offs between them?

Consider studying a problem (proving a lower bound and designing a matching algorithm) with respect to two metrics A and B. A standard approach in such cases is to consider results with different trade-offs between the two metrics. As a result, we know how hard it is to solve the problem for some fixed values of the metrics A and B but usually not for a general combination of A and B. Instead, consider a combined metric, say $A + B$ (more generally, a function of A and B), and suppose we show (with matching algorithm and lower bound) that there is some value of $A + B$ that is necessary and sufficient to solve the problem. Now, by simply varying the

amount each metric contributes to the compound metric, we automatically have a smooth scale of values of the metrics A and B for which the problem is solvable. Clearly, lower bounds and algorithms for such a combined metric are harder to prove than results for fixed values of the metrics; however, they also provide us with much better understanding of the nature of the problem.

Examples from biology can be used as inspiration and motivation for designing combined metrics that suit well-known computer science problems better than standard single metrics. In evolutionary terms, when an individual mutates, its fitness is evaluated based on some compound function of all its (newly developed and old) traits. Considering complex metrics that capture all these traits together can give us better understanding of the problem at hand and the necessary steps to solve it efficiently. In our foraging work, we identify the *selection metric* as the function that combines two different metrics (memory and probability range) and fits the problem well. A possible conclusion is that in this setting the selection metric is the "right way" to combine an individual's traits in order to evaluate its fitness.

5.1.2 New Models and Lower Bounds

Another standard approach in distributed computing theory is to treat a mathematical model as a fixed entity that can be fully "stretched and exploited" by the lower bounds and algorithms. Results are considered desirable and tight if the algorithms make use of every single capability provided by the model, and the lower bounds make use of every restriction in the capabilities of the computing entities. Such an approach is influenced by the strict mathematical formalism that underpins theoretical computer science, but it makes results less relevant to real engineering and biological systems.

In our house hunting work, we encountered an example of this situation where we have a model together with matching lower bound and algorithm. However, the algorithm uses the capabilities outlined in the model in very unnatural ways (uncharacteristic of any real biological system), which results in a fragile algorithm that does not perform correctly under the slightest noise in the environmental parameters

being modeled. One remedy to this issue is to change the model to reflect the noisy behaviors we want to consider. A potential risk in this approach is that the resulting model is so specific that it fits only a small class of problems, and is too involved to design and analyze algorithms mathematically.

We argue that in such situations, it is important to treat the model as a flexible entity and strive for simple, natural and robust algorithms. As an example, Algorithm 7 may not be optimal but it is resilient to perturbations of the parameters of the algorithm, and its correctness is not critically dependent on any one particular system model assumption. The only potential drawback of such algorithms is that they shift the complexity from the algorithm to the analysis of the correctness and efficiency of the algorithm. However, this is not necessarily a disadvantage considering how much easier it is to implement and maintain such algorithms in practice at the one-time cost of a more complicated mathematical proof.

5.1.3 Robust and Simple Algorithms

Finally, we elaborate some more on the specific robustness characteristics of algorithms that we consider desirable. All the models and algorithms presented in this thesis are as simple as possible, each exhibiting different robustness properties. Next, we list some of these properties.

- **No communication, or extremely limited communication.** The assumption that agents in a distributed system cannot communicate, or at least cannot communicate directly with one another, poses many difficulties in algorithm design and sidesteps the usual choice between message passing and shared memory models. Although such a model is not suitable for many distributed computing applications where agents/nodes have to exchange some information, it does allow algorithm designers to ignore an entire set of potential issues like message delays, implementing reliable channels, distinguishing between slow messages and crashed nodes, etc. All of these issues are really vulnerabilities of the algorithms with respect to uncertainty in the environment that make

algorithms less robust in real-world (engineering or biological) applications.

Our results suggest that limiting communication, or (if possible) removing communication altogether, brings algorithms one step closer to natural robustness and resilience against real-world perturbations. Our foraging work assumes absolutely no communication and shows that a general problem like searching the plane is solvable in optimal time with very limited other resources (selection metric) without having to rely on sending and receiving large and complicated messages between the searchers.

Our house-hunting work does use some communication in the form of tandem runs (recruitment), however, it is extremely rudimentary and allows for only small amount of information to be exchanged between communicating agents. The results confirm that even such limited communication is sufficient to solve consensus, also as evidenced by real ants in nature. Models of limited communication have already attracted the attention of researchers working on population protocols and the stone age model [48] where they solve traditionally difficult problems like consensus, leader election, MIS, and other graph problems. The density estimation algorithm in [83] uses only the encounter rates between agents as communication, which is suggested to be the case with many ant species [59].

Finally, our work on task allocation employs a new form of indirect communication between agents, using the environment as a medium. While this is reminiscent of the shared-memory type of communication, here the agents cannot read an write arbitrary bits of information. Instead, the environment provides each agent with probabilistic and approximate information about the current state of the system. Learning how to solve problems under such indirect communication models is challenging but it can save the algorithm designer from issues like atomicity properties of the shared-memory registers, corrupted registers, or Byzantine agents writing malicious information.

- **Approximate, probabilistic, and limited environment input.** In dis-

tributed computing theory, we often make assumptions on the amount of information agents have about the state of the system. For example, do agents know the total number of agents, do they know the diameter of the graph, do they have a common notion of time, etc. When we tackle a new problem, we usually start by assuming agents have all the information they need to solve the problem, and then we work on removing these assumptions one by one. Sometimes, of course, some information is critical to solving a problem, so we usually state, through a lower bound or an impossibility result, that the problem is unsolvable or hard to solve without some specific knowledge of some parameter. One such example, as we mentioned in Chapter 1 is designing a foraging algorithm that works without knowledge of the total number n of foragers. A lower bound [50] states that without knowing n, we have to pay a $\log n$ factor in the running time of the algorithm.

Our house-hunting work suggests that it is also worth answering the question of how well an algorithm performs with approximate information of the parameter of choice. This assumption can be thought of as an intermediate step between knowing the precise value of the parameter and not having any knowledge of it. We show that, under the right assumptions, not knowing the exact number of ants at a candidate nest does not affect the correctness and efficiency of the house-hunting algorithm significantly. In fact, assuming approximate knowledge of environmental parameters is a standard assumption in biology and other life sciences, which takes our house-hunting algorithm one step closer to being relevant to real-world ant colonies. We conjecture that making similar assumptions in theoretical distributed computing models can result in simpler algorithms that are easier to maintain under various fluctuations of the environment, and may also lead to new tools and techniques for interesting theoretical analysis.

- **Simple algorithms, composed of a single rule.** In recent years, we have seen a large number of complex algorithms in distributed computing theory that involve dozens of lines of pseudocode, attempting to address every single possible

event that can occur in the system, and meticulously listing the steps needed to react to that event. The resulting algorithms are not only hard to read, understand, and analyze, but they are usually fragile in terms of considering all the possible combinations of states in which each of the agents may be with respect to the current state of the environment. For these algorithms, it is usually difficult to prove correctness in an asynchronous environment where at any given point in time any agent may be executing any step of the complex algorithm. For the same reasons, dealing with faults can also be challenging for such algorithms.

Most of the algorithms in this thesis (perhaps with the exception of the optimal house-hunting algorithm) involve either a single rule, or a few simple rules, that each agent executes in each round (without knowledge of the round number). Even if we consider an asynchronous execution, we still know at any given point in time what step of its algorithm each agent is executing. Our results demonstrate that even such simple algorithms can solve difficult problems correctly and efficiently, shifting the burden of complexity from the algorithm to the mathematical analysis. The main advantage of such lightweight algorithms is that they tolerate faults in a natural way simply since all the agents are always performing the same kind of operation; even if some of them crash, other identical agents will take their place. Furthermore, combining algorithm simplicity with the lack of complex communication, we also have an easier way of dealing with asynchrony: all agents are executing the same rule and they do not depend on hearing from each other, so the exact time frame in which each agent executes the rule is less relevant.

- **New robustness property of randomized algorithms.** In our house hunting work, we considered a variation of the model by introducing uncertainty in the population estimates of ants. While this uncertainty model is bio-inspired (real ants are not believed to count precisely), it introduces the idea of changing the system model so that the probabilities used in a randomized algorithm are

perturbed by adversaries of variable strengths. For example, suppose some action is performed with probability p in the algorithm. In the new system model, we assume that this action is performed with probability p' such that for some $\epsilon \in (0, 1)$, $p' \in [(1 - \epsilon)p, (1 + \epsilon)p]$, and the probability p' may be chosen adversarily in the given range. Alternatively, we can assume that for some $\delta \in (0, 1)$, p' is in the given range with probability at least $1 - \delta$ and it is sampled from a distribution satisfying certain properties (for example, the expected value of p' is p). These different assumptions of the uncertainty models lead to different properties of the resulting randomized algorithms, potentially affecting their correctness and running times. To our knowledge, our house hunting work is the first to introduce this type of uncertainty.

We believe this style of uncertainty properties is important in making randomized distributed algorithms more relevant to both biological and engineering systems. In practical systems, randomized algorithms are implemented by using pseudorandom number generators that provide approximations of the probabilities used in the algorithms. In order to understand the correctness and efficiency guarantees of the resulting algorithms, it is crucial to understand how the potentially small (adversarial or probabilistic) perturbations of the probabilities affect the algorithms. In biological systems, individuals are believed to have even more limited access to randomization and less accurate estimates of real-world parameters, which imply even larger perturbations of the intended probabilities used in the algorithms. Thus, in order to understand the algorithms that evolved to solve various problems, we need to be able to design and analyze algorithms that are resilient to such uncertainty.

5.2 Lessons for Evolutionary Biologists

Biologists working on understanding social insect colonies are constantly baffled by observing behavior that is either not explained by current hypotheses, or contrary to existing hypotheses. One example of such behavior, as mentioned in Chapter 1, is

the existence of idle ants in an ant colony in the presence of unsatisfied tasks. The general structure of tackling such issues is first forming hypotheses and then testing these hypotheses through theoretical models or practical experiments. We believe we, as theoretical computer scientists, can help biologists in both of these steps.

Our work on task allocation is a great example of how theoretical results can help biologists generate a hypothesis about a specific ant colony behavior. By designing general and abstract models of task allocation and analyzing the resulting processes, we identified the ant-to-work ratio (together with the total amount of work needed) as the key parameter that determines the efficiency of task allocation, and that can explain the existence of idle ants in the colony (higher ant-to-work ratio implies both more efficient task allocation and more idle ants). This specific parameter has not been included in previous biological models of task allocation and has not even been measured experimentally in real ant colonies. While this hypothesis is not supported by any empirical findings yet, we believe our results are a good start to at least attempt a new direction in understanding the task allocation process in general and the idle ants phenomenon in particular.

In conclusion, we believe biologists can benefit from considering tools and techniques from distributed computing. The analysis of distributed algorithms can help biologists generate new hypotheses about observed ant behavior, and distributed computing models can help test and verify other existing hypotheses. Biologists have already shown interest in some distributed models of insect colonies [60, 86]. These models are usually continuous and rely on solving and analyzing complex differential equations; we hope that examples like our work on task allocation will encourage biologists to try simpler discrete models and techniques from theoretical distributed computing and complexity analysis. Finally, we hope that these new models and hypotheses about ant behavior can be verified with experiments and eventually lead to new discoveries about the behavior of ants and the reasons for this behavior.

Appendix A

Mathematical Preliminaries

This appendix includes basic mathematical results that we use throughout the earlier chapters.

A.1 Basic Probability Definitions and Results

In this section we state standard results from probability theory including some well-known concentration bounds. First, we show how to bound the expected value of a minimum in terms of the minimum of expected values.

Lemma A.1.1. *For each $k \geq 1$, let $I_1, \cdots, I_k$ be identically distributed independent binary random variables, and let $X = \sum_{i=1}^{k} I_i$. For an arbitrary constant $c > 0$:*

$$\mathbb{E}[\min\{X, c\}] \geq \frac{1}{2} \cdot \min\{\lfloor \mathbb{E}[X] \rfloor, c\}.$$

Proof. Let m be the median of X. By definition, $\Pr[X \geq m] \geq 1/2$. Since X is a binomial random variable, $m \geq \lfloor \mathbb{E}[X] \rfloor$ [72]. Let $m' = \min\{\lfloor \mathbb{E}[X] \rfloor, c\}$, so we have $m \geq m'$ and $\Pr[X \geq m'] \geq 1/2$.

$$\mathbb{E}[\min\{X, c\}] \geq \mathbb{E}[\min\{X, m'\}] \geq \Pr[X \geq m'] \cdot m' \geq \frac{m'}{2} = \frac{1}{2} \cdot \min\{\lfloor \mathbb{E}[X] \rfloor, c\}.$$

□

Corollary A.1.2. *For each $k \geq 1$, let $I_1, \cdots, I_k$ be identically distributed independent binary random variables, and let $X = \sum_{i=1}^{k} I_i$. For an arbitrary constant $c \geq 1$:*

$$\mathbb{E}[\min\{X, c\}] \geq \frac{1}{4} \cdot \min\{\mathbb{E}[X], c\}.$$

Proof. If $\mathbb{E}[X] \geq 1$, then $\lfloor \mathbb{E}[X] \rfloor \geq \mathbb{E}[X]/2$ and the corollary holds by Lemma A.1.1. If $\mathbb{E}[X] < 1$ and $\mathbb{E}[I_i] = p$ for each $1 \leq i \leq k$, it follows that $kp = \mathbb{E}[X] < 1$.

$$\begin{aligned}
\mathbb{E}[\min\{X, c\}] \geq \mathbb{E}[\min\{X, 1\}] \geq \Pr[X = 1] &= \sum_{i=1}^{k} p(1-p)^{(k-1)} \\
&= p \sum_{i=1}^{k} e^{-1} && \text{since } kp < 1 \\
&\geq e^{-1} \cdot \mathbb{E}[X] \\
&\geq \frac{1}{4} \cdot \mathbb{E}[X] \\
&\geq \frac{1}{4} \cdot \min\{\mathbb{E}[X], c\}.
\end{aligned}$$

□

Next, we state some well-known concentration bounds.

Theorem A.1.3 (Reverse Markov bound)**.** *Let X be an arbitrary random variable such that $X \leq B$ for some $B \in \mathbb{R}$. Then, for each $a \in \mathbb{R}$ and $a < B$:*

$$\Pr[X \leq a] \leq \frac{\mathbb{E}[B - X]}{B - a}.$$

Theorem A.1.4 (Chernoff bound)**.** *Let $X_1, \cdots, X_k$ be independent random variables such that for $1 \leq i \leq k$, $X_i \in \{0, 1\}$. Let $X = X_1 + X_2 + \cdots + X_k$ and let $\mu = \mathbb{E}[X]$. Then, for any $0 \leq \delta \leq 1$, it is true that:*

$$\begin{aligned}
\Pr[X > (1+\delta)\mu] &\leq e^{-\delta^2 \mu/2}, \\
\Pr[X < (1-\delta)\mu] &\leq e^{-\delta^2 \mu/3}, \\
\Pr[|X - \mu| > \delta\mu] &\leq 2e^{-\delta^2 \mu/3}.
\end{aligned}$$

Theorem A.1.5 (Two-sided Chernoff bound)**.** *Let $X_1, \cdots, X_k$ be independent random variables such that for $1 \leq i \leq k$, $X_i \in \{0, 1\}$. Let $X = X_1 + X_2 + \cdots + X_k$ and let $\mu = \mathbb{E}[X]$. Then, for any $0 \leq \delta \leq 1$, it is true that:*

$$\Pr[|X - \mu| > \delta\mu] \leq 2e^{-\delta^2\mu/3}$$

Theorem A.1.6. *Let $X_1, \cdots, X_n$ be arbitrary binary random variables. Also, let $X_1^*, \cdots, X_n^*$ be random variables that are mutually independent and such that for all i, X_i^* is independent of $X_1, \cdots, X_{i-1}$. Assume that for all i and all $x_1, \cdots, x_{i-1} \in \{0, 1\}$,*

$$P[X_i = 1 | X_1 = x_1, \cdots, X_{i-1} = x_{i-1}] \geq P[X_i^* = 1].$$

Then, for all $k \geq 0$, we have,

$$\Pr\left[\sum_{i=1}^{n} X_i < k\right] \leq P\left[\sum_{i=1}^{n} X_i^* < k\right],$$

and the latter term can be bounded by Chernoff bounds for independent random variables.

Theorem A.1.7 (Reverse Chernoff bound)**.** *Let $X_1, \cdots, X_k$ be independent random variables such that for $1 \leq i \leq k$, $X_i \in \{0, 1\}$. Let $X = X_1 + X_2 + \cdots + X_k$, let $\mu = \mathbb{E}[X]$, and let $p_i = \Pr[X_i = 1]$. If $p_i \leq 1/4$ for all $i \in [1, k]$, then for any $t > 0$, it is true that:*

$$\Pr[X - \mu > t] \geq \left(\frac{1}{4}\right) e^{-2t^2/\mu}.$$

Theorem A.1.8 (Paley-Zygmunt inequality [87])**.** *Let $X \geq 0$ be a random variable with finite variance. For each $0 \leq \theta \leq 1$:*

$$\Pr[X > \theta\mathbb{E}[X]] \geq (1 - \theta)^2 \left(\frac{\mathbb{E}[X]^2}{\mathbb{E}[X^2]}\right).$$

A.2 Markov Chains

In this section, we state some basic results on Markov chains.

Theorem A.2.1 (Feller [52]). *In an irreducible Markov chain with period t the states can be divided into t mutually exclusive classes $G_0, \cdots, G_{t-1}$ such that it is true that (1) if $s \in G$ then the probability of being in state s in some round $r \geq 1$ is 0 unless $r = \tau + vt$ for some $v \in \mathbb{N}$, and (2) a one-step transition always leads to a state in the right neighboring class (in particular from G_{t-1} to G_0). In the chain with matrix P^t each class G corresponds to an irreducible closed set.*

The next theorem establishes a bound on the difference between the stationary distribution of a Markov chain and the distribution resulting after k steps.

Lemma A.2.2 (Rosenthal [101]). *Let $P(x, \cdot)$ be the transition probabilities for a time-homogeneous Markov chain on a general state space $\mathcal{X}$. Suppose that for some probability distribution $Q(\cdot)$ on $\mathcal{X}$, some positive integers k and k_0, and some $\epsilon > 0$,*

$$\forall x \in \mathcal{X} : P^{k_0}(x, \cdot) \geq \epsilon Q(\cdot),$$

where P^{k_0} represents the k_0-step transition probabilities. Then for any initial distribution π_0, the distribution π_k of the Markov chain after k steps satisfies

$$\|\pi_k - \pi\| \leq (1 - \epsilon)^{\lfloor k/k_0 \rfloor},$$

where $\|\cdot\|$ is the ∞-norm and π is any stationary distribution. (In particular, the stationary distribution is unique.)

The next result uses general results from number theory to bound the lengths of paths in a Markov chain.

Lemma A.2.3. *In any irreducible, aperiodic Markov chain with $|S|$ states, there exists an integer $k \leq 2|S|^2$ such that there is a walk of length k between any pair of states in the Markov chain.*

Proof. By the definition of periodicity, for each state of the Markov chain, it is true that the greatest common divisor of the lengths of all the cycles that pass through that state is 1. Let the total number of distinct cycles in the Markov chain be m and let $(a_1, \cdots, a_m)$ denote the lengths of these cycles where $a_1 \leq \cdots \leq a_m$. The Frobenius number $F(a_1, \cdots, a_m)$ of the sequence $(a_1, \cdots, a_m)$ is the largest integer such that it is not possible to express it as a linear combination of $(a_1, \cdots, a_m)$ and non-negative integer coefficients. By a simple bound on the Frobenius number [18], we know that $F(a_1, \cdots, a_m) \leq (a_1 - 1)(a_2 - 1) - 1$. Since a_1 and a_2 refer to cycle lengths in our Markov chain we know that $a_1, a_2 \leq |S|$. So, it is true that $F(a_1, \cdots, a_m) \leq |S|^2$ and we can express every integer greater than $F(a_1, \cdots, a_m)$ as a non-negative integer linear combination of $(a_1, \cdots, a_m)$.

Let i and j be arbitrary states in the Markov chain and let $d(i, j)$ be the shortest path between i and j. Let $k = 2|S|^2$. By the argument above, we know that there is a walk starting at state i and ending at state i of length $k - d(i, j) \geq |S|^2$. Appending the shortest path between i and j to the end of that walk results in a walk from i to j of length exactly k. □

Bibliography

[1] Yehuda Afek, Noga Alon, Omer Barad, Eran Hornstein, Naama Barkai, and Ziv Bar-Joseph. A biological solution to a fundamental distributed computing problem. *Science*, 331(6014):183–185, 2011.

[2] Carlos Aguirre, Jaime Martinez-Muñoz, Fernando Corbacho, and Ramón Huerta. Small-world topology for multi-agent collaboration. In *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, pages 231–235. IEEE, 2000.

[3] Susanne Albers and Monika R. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164–1188, 2000.

[4] Noga Alon, Chen Avin, Michal Kouckỳ, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. Many random walks are faster than one. *Combinatorics, Probability and Computing*, 20(04):481–502, 2011.

[5] Christoph Ambuhl, Leszek Gasieniec, Andrzej Pelc, Tomasz Radzik, and Xiaohui Zhang. Tree exploration with logarithmic memory. *ACM Transactions on Algorithms*, 7(2):17, 2011.

[6] Dana Angluin, James Aspnes, Michael J. Fischer, and Hong Jiang. Self-stabilizing population protocols. In *Principles of Distributed Systems*, pages 103–117. Springer, 2005.

[7] Michal Arbilly, Uzi Motro, Marcus W. Feldman, and Arnon Lotem. Co-evolution of learning complexity and social foraging strategies. *Journal of Theoretical Biology*, 267(4):573–581, 2010.

[8] James Aspnes and Eric Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer, 2009.

[9] Karl Johan Aström and Richard M. Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton University Press, 2010.

[10] Hagit Attiya and Jennifer Welch. *Distributed computing: fundamentals, simulations, and advanced topics*. John Wiley & Sons, 2004.

[11] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Riccardo Silvestri. Plurality consensus in the gossip model. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 371–390. SIAM, 2015.

[12] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, Riccardo Silvestri, and Luca Trevisan. Simple dynamics for plurality consensus. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 247–256. ACM, 2014.

[13] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Stabilizing consensus with many opinions. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 620–635. SIAM, 2016.

[14] Michael A. Bender, Antonio Fernández, Dana Ron, Amit Sahai, and Salil Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 269–278. ACM, 1998.

[15] Samuel N. Beshers and Jennifer H. Fewell. Models of division of labor in social insects. *Annual review of entomology*, 46(1):413–440, 2001.

[16] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 263(1376):1565–1569, 1996.

[17] Vincenzo Bonifaci, Kurt Mehlhorn, and Girish Varma. Physarum can compute shortest paths. *Journal of Theoretical Biology*, 309:121–133, 2012.

[18] Alfred Brauer. On a problem of partitions. *American Journal of Mathematics*, 64(1):299–312, 1942.

[19] Scott Camazine. *Self-organization in biological systems*. Princeton University Press, 2003.

[20] Scott Camazine. Self-organizing systems. *Encyclopedia of Cognitive Science*, 2006.

[21] T.T. Cao and A. Dornhaus. Ants use pheromone markings in emigrations to move closer to food-rich areas. *Insectes Sociaux*, 59(1):87–92, 2012.

[22] D. Charbonneau and A. Dornhaus. Workers 'specialized' on inactivity: behavioral consistency of inactive workers and their role in task allocation. *Behavioral Ecology and Sociobiology*, published online, 2015.

[23] D. Charbonneau, N. Hillis, and A. Dornhaus. 'lazy' in nature: ant colony time budgets show high 'inactivity' in the field as well as in the lab. *Insectes Sociaux*, 62(1):31–35, 2015.

[24] D. Charbonneau, H. Nguyen, M. C. Shin, and A. Dornhaus. Who are the 'lazy' ants? Concurrently testing multiple hypotheses for the function of inactivity in social insects. *Scientific Reports*, submitted.

[25] Daniel Charbonneau and Anna Dornhaus. When doing nothing is something. How task allocation mechanisms compromise between flexibility, efficiency, and inactive agents. *Journal of Bioeconomics*, 17:217–242, 2015.

[26] Daniel Charbonneau, Neil B. Hillis, and Anna Dornhaus. Are 'lazy' ants selfish? Testing whether highly inactive workers invest more in their own reproduction than highly active workers. submitted.

[27] Fan Chung, Shirin Handjani, and Doug Jungreis. Generalizations of Polya's urn problem. *Annals of Combinatorics*, 7(2):141–153, 2003.

[28] A. Cornejo, A. R. Dornhaus, N. A. Lynch, and R. Nagpal. Task allocation in ant colonies. In *Proceedings of the 28th Symposium on Distributed Computing (DISC)*, pages 46–60, 2014.

[29] Stefanie M. Countryman, Martin C. Stumpe, Sam P. Crow, Frederick R. Adler, Michael J. Greene, Merav Vonshak, and Deborah M. Gordon. Collective search by ants in microgravity. *Frontiers in Ecology and Evolution*, 3, 2015.

[30] Adam L. Cronin. Synergy between pheromone trails and quorum thresholds underlies consensus decisions in the ant Myrmecina nipponica. *Behavioral Ecology and Sociobiology*, 67(10):1643–1651, 2013.

[31] Gautham P. Das, Thomas M. McGinnity, Sonya A. Coleman, and Laxmidhar Behera. A distributed task allocation algorithm for a multi-robot system in healthcare facilities. *Journal of Intelligent & Robotic Systems*, 80(1):33–58, 2015.

[32] Partha Dasgupta, Zvi M. Kedem, and Michael O. Rabin. Parallel processing on networks of workstations: A fault-tolerant, high performance approach. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, pages 467–474. IEEE, 1995.

[33] Mathijs De Weerdt, Yingqian Zhang, and Tomas Klos. Distributed task allocation in social networks. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, page 76. ACM, 2007.

[34] Xiaotie Deng and Christos H. Papadimitriou. Exploring an unknown graph. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 355–361. IEEE, 1990.

[35] Krzysztof Diks, Pierre Fraigniaud, Evangelos Kranakis, and Andrzej Pelc. Tree exploration with little memory. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 588–597. Society for Industrial and Applied Mathematics, 2002.

[36] Benjamin Doerr, Leslie Ann Goldberg, Lorenz Minder, Thomas Sauerwald, and Christian Scheideler. Stabilizing consensus with the power of two choices. In *Proceedings of the 23rd annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 149–158. ACM, 2011.

[37] Matina C. Donaldson-Matasci, Gloria DeGrandi-Hoffman, and Anna Dornhaus. Bigger is better: honeybee colonies as distributed information-gathering systems. *Animal Behaviour*, 85(3):585–592, 2013.

[38] Joseph Leo Doob. *Stochastic processes*. John Wiley & Sons, 1990.

[39] A. Dornhaus and N.R. Franks. Colony size affects collective decision-making in the ant temnothorax albipennis. *Insectes Sociaux*, 53(4):420–427, 2006.

[40] Anna Dornhaus. Specialization does not predict individual efficiency in an ant. *PLoS Biology*, 6(11):e285, 2008.

[41] Anna Dornhaus. Finding optimal collective strategies using individual-based simulations: colony organization in social insects. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):25–37, 2012.

[42] Anna Dornhaus, Jo-Anne Holley, Victoria G. Pook, Gemma Worswick, and Nigel R. Franks. Why do not all workers work? colony size and workload during emigrations in the ant temnothorax albipennis. *Behavioral Ecology and Sociobiology*, 63(1):43–51, 2008.

[43] Anna Dornhaus, Nancy Lynch, Tsvetomira Radeva, and Hsin-Hao Su. Brief announcement: Distributed task allocation in ant colonies. In *Proceedings of the 29th International Symposium on Distributed Computing*, pages 657–658. Springer, 2015.

[44] Anna Dornhaus, Scott Powell, and Sarah Bengston. Group size and its effects on collective organization. *Annual review of entomology*, 57:123–141, 2012.

[45] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. The communication complexity of distributed task allocation. In *Proceedings of the 31st ACM Symposium on Principles of Distributed Computing*, pages 67–76. ACM, 2012.

[46] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.

[47] Yuval Emek, Tobias Langner, Jara Uitto, and Roger Wattenhofer. Solving the ANTS problem with asynchronous finite state machines. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming*, pages 471–482, 2014.

[48] Yuval Emek and Roger Wattenhofer. Stone age distributed computing. In *Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing*, pages 137–146. ACM, 2013.

[49] Theodore A. Evans. Foraging and building in subterranean termites: task switchers or reserve labourers? *Insectes Sociaux*, 53(1):56–64, 2006.

[50] Ofer Feinerman and Amos Korman. Memory lower bounds for randomized collaborative search and implications for biology. In *Distributed Computing*, pages 61–75. Springer, 2012.

[51] Ofer Feinerman, Amos Korman, Zvi Lotker, and Jean-Sébastien Sereni. Collaborative search on the plane without communication. In *Proceedings of the 31st ACM Symposium on Principles of Distributed Computing*, pages 77–86. ACM, 2012.

[52] Willliam Feller. *An introduction to probability theory and its applications*. John Wiley & Sons, 2008.

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

[54] Pierre Fraigniaud, Leszek Gasieniec, Dariusz R. Kowalski, and Andrzej Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006.

[55] N. R. Franks, A. Dornhaus, J. A. R. Marshall, and F.-X. Dechaume-Moncharmont. The dawn of a golden age in mathematical insect sociobiology. *Organization of Insect Societies: From Genome to Sociocomplexity*, pages 437–459, 2009.

[56] Chryssis Georgiou. *Do-All Computing in Distributed Systems: Cooperation in the Presence of Adversity*. Springer Science & Business Media, 2007.

[57] Mohsen Ghaffari, Cameron Musco, Tsvetomira Radeva, and Nancy Lynch. Distributed house-hunting in ant colonies. In *Proceedings of the 34th ACM Symposium on Principles of Distributed Computing*, PODC, pages 57–66. ACM, 2015.

[58] Luc-Alain Giraldeau and Thomas Caraco. *Social foraging theory*. Princeton University Press, 2000.

[59] Deborah M. Gordon. *Ant encounters: interaction networks and colony behavior*. Princeton University Press, 2010.

[60] Deborah M. Gordon, Brian C. Goodwin, and L.E.H. Trainor. A parallel distributed model of the behaviour of ant colonies. *Journal of Theoretical Biology*, 156(3):293–307, 1992.

[61] Michael J. Greene and Deborah M. Gordon. Interaction rate informs harvester ant task decisions. *Behavioral Ecology*, 18(2):451–455, 2007.

[62] C. Groh, J. Tautz, and W. Rossler. Synaptic organization in the adult honey bee brain is influenced by brood-temperature control during pupal development. *PNAS*, 101(12):4268–4273, 2004.

[63] Jeremy Gunawardena. Models in biology: 'accurate descriptions of our pathetic thinking'. *BMC Biology*, 12(1):1–11, 2014.

[64] Hermann Haken. *Synergetics. An Introduction. Non-equilibrium Phase Trasitions and Self-Organization in Physics, Chemistry, and Biology*. Berlin, 1977.

[65] Christiane I.M. Healey and Stephen C. Pratt. The effect of prior experience on nest site evaluation by the ant Temnothorax curvispinosus. *Animal Behaviour*, 76(3):893–899, 2008.

[66] Joan M Herhers. Social organisation in leptothorax ants: within-and between-species patterns. *Psyche*, 90(4):361–386, 1983.

[67] K. Holder and G.A. Polis. Optimal and central-place foraging theory applied to a desert harvester ant, Pogonomyrmex californicus. *Oecologia*, 72(3):440–448, 1987.

[68] William O.H. Hughes, Seirian Sumner, Steven Van Borm, and Jacobus J. Boomsma. Worker caste polymorphism has a genetic basis in acromyrmex leaf-cutting ants. *Proceedings of the National Academy of Sciences*, 100(16):9394–9397, 2003.

[69] Jennifer M. Jandt and Anna Dornhaus. Competition and cooperation: bumblebee spatial organization and division of labor may affect worker reproduction late in life. *Behavioral Ecology and Sociobiology*, 65:2341–2349, 2011.

[70] Jennifer M. Jandt and Anna Dornhaus. Bumblebee response thresholds and body size: does worker diversity increase colony performance? *Animal Behaviour*, 87:97–106, 2014.

[71] Jennifer M. Jandt, Eden Huang, and Anna Dornhaus. Weak specialization of workers inside a bumble bee (Bombus impatiens) nest. *Behavioral Ecology and Sociobiology*, 63(12):1829–1836, 2009.

[72] Rob Kaas and Jan M. Buhrman. Mean, median and mode in binomial distributions. *Statistica Neerlandica*, 34(1):13–18, 1980.

[73] Richard Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vocking. Randomized rumor spreading. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 565–574. IEEE, 2000.

[74] Eric Korpela, Dan Werthimer, David Anderson, Jeff Cobb, and Matt Lebofsky. SETI HOME – massively distributed computing for SETI. *Computing in Science & Engineering*, 3(1):78–83, 2001.

[75] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, 16(2):133–169, 1998.

[76] Christoph Lenzen, Nancy Lynch, Calvin Newport, and Tsvetomira Radeva. Trade-offs between selection complexity and performance when searching the plane without communication. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing*, pages 252–261. ACM, 2014.

[77] Christoph Lenzen, Nancy Lynch, Calvin Newport, and Tsvetomira Radeva. Searching without communicating: tradeoffs between performance and selection complexity. *Distributed Computing*, pages 1–23, 2016.

[78] Nancy A. Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[79] E. Mallon, S. Pratt, and N. Franks. Individual and collective decision-making during nest site selection by the ant leptothorax albipennis. *Behavioral Ecology and Sociobiology*, 50(4):352–359, 2001.

[80] Eamonn B. Mallon and Nigel R. Franks. Ants estimate area using Buffon's needle. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 267(1445):765–770, 2000.

[81] M. A. McLeman, S. C. Pratt, and N. R. Franks. Navigation using visual landmarks by the ant Leptothorax albipennis. *Insectes Sociaux*, 49(3):203–208, 2002.

[82] Charles D. Michener. Reproductive efficiency in relation to colony size in hymenopterous societies. *Insectes Sociaux*, 11(4):317–341, 1964.

[83] Cameron Musco, Hsin-Hao Su, and Nancy Lynch. Ant-inspired density estimation via random walks. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 469–478. ACM, 2016.

[84] Casey O'Brien. *Solving ANTS with Loneliness Detection and Constant Memory*. MEng Thesis, MIT EECS Department, 2014.

[85] S. OḊonnell and S. J. Bulova. Worker connectivity: a review of the design of worker communication systems and their effects on task performance in insect societies. *Insectes Sociaux*, 54(3):203–210, 2007.

[86] S. W. Pacala, D. M. Gordon, and H. C. J. Godfray. Effects of social group size on information transfer and task allocation. *Evolutionary Ecology*, 10(2):127–165, 1996.

[87] R. E. A. C. Paley and A. Zygmund. On some series of functions, (3). *Mathematical Proceedings of the Cambridge Philosophical Society*, 28(2):190–205, 1932.

[88] Petrişor Panaite and Andrzej Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33(2):281–295, 1999.

[89] David Peleg. Distributed computing. *SIAM Monographs on Discrete Mathematics and Applications*, 5, 2000.

[90] Henrique M. Pereira and Deborah M. Gordon. A trade-off in task allocation between sensitivity to the environment and response time. *Journal of Theoretical Biology*, 208(2):165–184, 2001.

[91] Evlyn Pless, Jovel Queirolo, Noa Pinter-Wollman, Sam Crow, Kelsey Allen, Maya B. Mathur, and Deborah M. Gordon. Interactions increase forager availability and activity in harvester ants. *PloS One*, 10(11):e0141971, 2015.

[92] Stephen C. Pratt. Quorum sensing by encounter rates in the ant Temnothorax albipennis. *Behavioral Ecology*, 16(2):488–496, 2005.

[93] Stephen C. Pratt. Nest site choice in social insects. In *Encyclopedia of Animal Behavior*, pages 534 – 540. Academic Press, Oxford, 2010.

[94] Stephen C. Pratt, Eamonn B. Mallon, David J. Sumpter, and Nigel R. Franks. Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant Leptothorax albipennis. *Behavioral Ecology and Sociobiology*, 52(2):117–127, 2002.

[95] Stephen C. Pratt and David J. T. Sumpter. A tunable algorithm for collective decision-making. *Proceedings of the National Academy of Sciences*, 103(43):15906–15910, 2006.

[96] Fabien Ravary, Emmanuel Lecoutey, Gwenaël Kaminski, Nicolas Châline, and Pierre Jaisson. Individual experience alone can generate lasting division of labor in ants. *Current Biology*, 17(15):1308–1312, 2007.

[97] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):17, 2008.

[98] Javier Retana and Xim Cerdá. Social organization of cataglyphis cursor ant colonies (Hymenoptera, Formicidae): Inter-, and intraspecific comparisons. *Ethology*, 84(2):105–122, 1990.

[99] Elva J. H. Robinson, Duncan E. Jackson, Mike Holcombe, and Francis L. W. Ratnieks. Insect communication: "no entry" signal in ant foraging. *Nature*, 438(7067):442–442, 2005.

[100] Gene E. Robinson. Regulation of division of labor in insect societies. *Annual Review of Entomology*, 37(1):637–665, 1992.

[101] Jeffrey S. Rosenthal. Rates of convergence for data augmentation on finite sample spaces. *The Annals of Applied Probability*, pages 819–839, 1993.

[102] Takao Sasaki and Stephen C. Pratt. Emergence of group rationality from irrational individuals. *Behavioral Ecology*, 22(2):276–281, 2011.

[103] Takao Sasaki and Stephen C. Pratt. Groups have a larger cognitive capacity than individuals. *Current Biology*, 22(19):R827–R829, 2012.

[104] Takao Sasaki and Stephen C. Pratt. Ants learn to rely on more informative attributes during decision-making. *Biology Letters*, 9(6):20130667, 2013.

[105] P. Schmid-Hempel. Reproductive competition and the evolution of work load in social insects. *The American Naturalist*, 135:501–526, 1990.

[106] T.D. Seeley. *Honeybee ecology. A study of adaptation in social life.* Princeton University Press, 1985.

[107] A. B. Sendova-Franks and N. R. Franks. Spatial relationships within nests of the ant Leptothorax unifasciatus (Latr.) and their implications for the division of labour. *Animal Behaviour*, 50(1):121–136, 1995.

[108] Ana B. Sendova-Franks, Rebecca K. Hayward, Benjamin Wulf, Thomas Klimek, Richard James, Robert Planqué, Nicholas F. Britton, and Nigel R. Franks. Emergency networking: famine relief in ant colonies. *Animal Behaviour*, 79(2):473–485, 2010.

[109] Maria R. Servedio, Yaniv Brandvain, Sumit Dhole, Courtney L. Fitzpatrick, Emma E. Goldberg, Caitlin A. Stern, Jeremy Van Cleve, and D. Justin Yeh. Not just a theory – the utility of mathematical models in evolutionary biology. *PLoS Biol*, 12(12):e1002017, 2014.

[110] D. Sumpter and S. Pratt. A modelling framework for understanding social insect foraging. *Behavioral Ecology and Sociobiology*, 53(3):131–144, 2003.

[111] Edward O. Wilson. Behavioral discretization and the number of castes in an ant species. *Behavioral Ecology and Sociobiology*, 1(2):141–154, 1976.

[112] Edward O. Wilson. Caste and division of labor in leaf-cutter ants (Hymenoptera: Formicidae: Atta). *Behavioral Ecology and Sociobiology*, 7(2):157–165, 1980.