Solving ANTS with Loneliness Detection and Constant Memory

by

Casey O'Brien

B.S., Computer Science, B.S., Mathematics
Massachusetts Institute of Technology (2014)

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

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2015

Author .....
Department of Electrical Engineering and Computer Science
August 17, 2015

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

Accepted by .....
Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Solving ANTS with Loneliness Detection and Constant Memory

by

Casey O'Brien

Submitted to the Department of Electrical Engineering and Computer Science
on August 17, 2015, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In 2012, Feinerman et al. introduced the *Ants Nearby Treasure Search* (ANTS) problem [1]. In this problem, k non-communicating agents with unlimited memory, initially located at the origin, try to locate a treasure distance D from the origin. They show that if the agents know k, then the treasure can be located in the optimal $O(D + D^2/k)$ steps. Furthermore, they show that without knowledge of k, the agents need $\Omega((D + D^2/k) \cdot \log^{1+\epsilon} k)$ steps for some $\epsilon > 0$ to locate the treasure. In 2014, Emek et al. studied a variant of the problem in which the agents use only constant memory but are allowed a small amount of communication [2]. Specifically, they allow an agent to read the state of any agent sharing its cell. In this paper, we study a variant of the problem similar to that in [2], but where the agents have even more limited communication. Specifically, the only communication is *loneliness detection*, in which an agent in able to sense whether it is the only agent located in its current cell. To solve this problem we present an algorithm Hybrid-Search, which locates the treasure in $O(D \cdot \log k + D^2/k)$ steps in expectation. While this is slightly slower than the straightforward lower bound of $\Omega(D + D^2/k)$, it is faster than the lower bound for agents locating the treasure without communication.

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

Acknowledgments

I would like to thank Mira Radeva for her help throughout the process of completing this thesis. Our discussions led to many of the results presented in this paper, and she helped me sort through many difficult proofs.

I would also like to thank my advisor, Professor Nancy Lynch, for the effort she spent guiding me through this thesis. Her help was instrumental in developing the work presented in this paper, especially in formalizing all the probabilistic proofs. Her comments on my many drafts helped me learn so much and were key in developing my arguments.

Finally, I would like to thank all the others who took the time to discuss my thesis with me and provide valueable insight, specifically Cameron Musco and Mohsen Ghaffari. I really appreciate the opportunity to have worked with this group.

Contents

List of Figures

Chapter 1

Introduction

Consider a colony of ants located at their nest. The colony needs to leave the nest and locate a food source. With their limited capabilities, the ants need to work collaboratively to locate the food in a timely manner. This problem has been studied in detail from the perspective of ant biologists. Recently, it has garnered attention from the perspective of distributed algorithmists.

In 2012, Feinerman et al. introduced the Ants Nearby Treasure Search (ANTS) problem [1]. In this problem, there are k non-communicating agents with unlimited memory, initially located at the origin of a two-dimensional grid. There is a treasure located a Manhattan distance D from the origin. The goal is for an agent to locate the treasure as quickly as possible, both in terms of D and k (refer to Section 2 for the formal model).

One of the reasons that this problem is interesting is its possible applications to biology. A handful of papers following [1] ([2], [3], [5]) studied the same problem using different models for the agents. However, none of the models that have been studied thus far properly represent ants in nature. In this paper, we propose a new model which more accurately represents the capabilities of ants.

It is well established that ants are able to sense the presence of other ants in their colony [4]. When an ant is within reach of another ant, it uses its antennae to sense the hydrocarbons on the back of the other ant. Each ant colony has a distinct scent, and so by doing this, the ant is able to determine if the other ant is a member of its

colony.

Consider a situation in which multiple nearby ant colonies are all performing an algorithm to locate a food source. When an ant is close to another ant, it can use its antennae to sense whether the other ant is a member of its colony. This would allow the ant to coordinate with only members of its own colony when performing the algorithm. Motivated by this ability, we study a model in which the only communication allowed by an agent is the ability to sense whether it is in the presence of another agent performing the algorithm. When there is only a single colony performing the algorithm, this ability is equivalent to sensing whether an agent is in the presence of any other agent.

1.1 Results

In this paper, we study a variant of the Ants Nearby Treasure Search (ANTS) problem, proposed by Feinerman et. al in 2012. In this problem, k agents navigate a two-dimensional grid, attempting to locate an adversarially placed treasure Manhattan distance D from the origin. All the agents begin at the origin. In a single round, an agent can move to the north, east, south, west, or remain in the same position [1].

It is straightforward to show that it takes $\Omega(D + D^2/k)$ rounds in expectation to locate the treasure [1]. Let T be the expected number of rounds for some algorithm to locate the treasure. Clearly $T \geq D$, because some agent has to reach the treasure. We also claim that $T \geq D^2/4k$. To see why, suppose we have $T < D^2/4k$. Rearranging yields $2Tk < D^2/2$. But this says that by after $2T$ rounds, more than half the cells have not been visited by any agent. Thus, there is some cell such that the probability that is it visited by some agent in $2T$ rounds is less than $1/2$. However, if the treasure is placed there, then the expected number of rounds to find the treasure is greater than T, yielding a contradiction.

In our model, each agent possesses only constant memory. Additionally, the agents are allowed very limited communication. Specifically, we allow our agents to have *loneliness detection* [6], which allows them to sense whether or not they are the only

agent at their current position. Our agents also have the ability to sense whether or not their current position is the origin.

Throughout this paper, we consider only a single set of agents searching for the treasure. However, we note that if we were to consider the case mentioned in the previous section, where multiple sets of agents were performing the algorithm at once from different starting locations, the obvious generalization of loneliness detection would be the ability to detect whether an agent was sharing a cell with another agent in its own set. If we were to substitute this ability for loneliness detection, then all the results in this paper would still be valid, even with multiple sets of agents performing the algorithm separately.

We present an algorithm Hybrid-Search which is able to locate the treasure in $O(D \cdot \log k + D^2/k)$ rounds in expectation in this new model. The algorithm works by having the agents randomly decide whether to execute Geometric-Search, which locates the treasure in $O(D)$ rounds with high probably if $D < (\log k)/2$, and Rectangle-Search, which locates the treasure in $O(D \cdot \log k + D^2/k)$ rounds in expectation if $D = \Omega(\log k)$. Furthermore, we conjecture that Rectangle-Search actually locates the treasure in $O(D \cdot \log\log k + D^2/k)$ if $D = \Omega(\log k)$. If this conjecture is true, then Hybrid-Search locates the treasure in $O(D \cdot \log\log k + D^2/k)$ rounds in expectation.

The algorithm Geometric-Search is a very simple protocol which does not involve any communication between the agents. In Rectangle-Search, the agents work in a carefully coordinated effort. Each agent consecutively executes three phases of the algorithm, and in each phase makes use of its loneliness detection ability.

We note that this algorithm has the optimal runtime when the treasure is far away (i.e., when $D = \Omega(k \log k)$). When the treasure is close to the origin ($D = o(k \log k)$), the algorithm has near optimal performance, locating the treasure in $O(D \cdot \log k)$ rounds in expectation.

1.2 Related Work

In [1], Feinerman et al. presented the original version of the Ants Nearby Treasure Search (ANTS) problem. In this version of the problem, the agents are unable to communicate, and are assumed to have infinite memory.

In the paper they present two algorithms. The first algorithm is non-uniform in k (i.e., assumes that the agents are aware of the value of k). The general idea behind the algorithm is that the agents randomly choose a cell and perform a *spiral search* around that cell. To perform a spiral search, the agent begins at some cell, and then walks in an outward spiral around that cell. We note that only an agent with infinite memory can perform spiral searches of arbitrary size. They show that this algorithm achieves the optimal runtime of $O(D + D^2/k)$.

The second algorithm they present is uniform in k (i.e., does not assume the agents have any knowledge of the value of k). The essence of this algorithm is that the agents guess k and then perform the previous algorithm, updating their estimated value of k if they fail to find the treasure based on their previous guess. This algorithm runs in $O((D + D^2/k) \cdot \log^{1+\epsilon} k)$ for some $\epsilon > 0$. Furthermore, they prove that this is optimal for any model in which the agents have no communication and no knowledge of k.

In [2], Emek et al. devised an optimal algorithm for the case where the agents have only finite memory. In their model, they allow the agents to have some communication. Specifically, an agent can read the state of any other agent sharing the same position.

The algorithm works by dividing the agents randomly between two algorithms. One algorithm is efficient when the treasure is far away. It locates the treasure in $O(D + D^2/k + \log k)$ with high probability. To do so, the agents split into search teams, and each search team is responsible for searching all the cells exactly d moves from the origin for some d. The teams begin by searching cells close to the origin, and then move outwards and search cells further way.

The other algorithm is efficient when the treasure is very close to the origin. It locates the treasure in $O(D)$ with high probability when there $D < (\log k)/2$. To

take advantage of the strengths of both algorithms, the agents randomly choose which algorithm to perform. This results in an overall runtime of $O(D + D^2/k)$ with high probability.

In [2], they also show how to modify the algorithm to work in the case where the agents move asynchronously. In [5], Langner et al. show how to modify the algorithm to be failure resistant. Specifically, the algorithm is modified so that it locates the treasure in $O(D+D^2/k+Df)$ with high probability, even if there are up to f failures.

1.3 Discussion

The algorithm presented in this paper is heavily inspired by the work in [2], so it is useful to compare our results to the results achieved there. The difference between the models is that their model allows much more communication than our model. Recall that their model allows an agent to read the states of all other agents in that cell. Our model simply allows an agent to sense whether another agent is present in its current cell. Both models assume that an agent is able to sense whether it is currently at the origin.

The algorithm in [2] is able to achieve the optimal runtime of $O(D + D^2/k)$ with high probability. The algorithm we present in this paper is able to achieve a runtime of $O(D \cdot \log k + D^2/k)$ in expectation, which is very close to the optimal runtime. In fact, we conjecture that the runtime is actually $O(D \cdot \log \log k + D^2/k)$ rounds in expectation.

1.4 Organization

In Chapter 2, we present the formal model for this paper. In Chapter 3, we present our algorithm Rectangle-Search. This algorithm locates the treasure efficiently when the treasure is located far from the origin. In Chapter 4, we present another new algorithm, Geometric-Search, which locates the treasure efficiently when it is located near the origin. We also describe the algorithm Hybrid-Search, which

uses both algorithms as subroutines, and achieves the near-optimal runtime of $O(D \cdot \log k + D^2/k)$ in expectation.

Chapter 2

Model

In our model, we consider the infinite two-dimensional grid $\mathbb{Z}^2$. The grid is explored by $k \in \mathbb{N}$ identical agents. Each agent is always located in some cell, and multiple agents may occupy the same cell. All agents are initially located at the origin. Agents are able to move north, east, south, or west. An agent is not aware of its current position in the grid, but can sense whether or not it is at the origin. We assume that each agent has access to a fair coin. The agents have only constant memory. Finally, the agents are capable of *loneliness detection* [6], that is, they can sense whether they are the only agent currently in their cell.

2.1 Agents

Each agent is modeled as a probabilistic finite state automaton. Each automaton is a tuple (S, s_0, δ, M), defined as follows:

1. S is a set of states.
2. $s_0 \in S$ is the unique start state.
3. $\delta : S \times \{\text{TRUE}, \text{FALSE}\} \times \{\text{TRUE}, \text{FALSE}\} \to \Pi$ is the state transition function, where Π is a set of probability distributions on S. It takes in a state $s \in S$, a boolean indicating if the agent is alone, and a boolean indicating if the agent is at the origin, and returns a probability distribution on S.

4. $M : S \to \{\text{NORTH}, \text{EAST}, \text{SOUTH}, \text{WEST}, \text{NONE}\}$ determines the moves the agent makes. It maps each state $s \in S$ to a move to be performed by the agent, where a move of NONE indicates staying in the same position.

We say that an agent is executing RECTANGLE-SEARCH if its *algorithm* variable is RECTANGLE, and similarly for GEOMETRIC-SEARCH. We say that an agent is in the *Separation Phase* if the *phase* variable of its current state is set to SEPARATION, and similarly for the *Allocation* and *Search Phase*s.

2.2 Executions

An execution of the algorithm consists of synchronous *rounds*. In a single round of the algorithm, all agents apply their M functions and move to new positions based on the results. Once all agents have completed their moves, they apply their δ functions to update their state.

We define *time* i to be the moment after i rounds have been completed. Let A be the set of agents. Then we define the *system configuration* $C_i : A \to S \times \mathbb{Z}^2$ for $i \geq 0$ to be a function which takes an agent and returns the state and position of the agent at time i. Formally, an execution of the algorithm is a sequence of system configurations $C_0, C_1, C_2, \ldots$, where for every agent a, $C_0(a) = (s_0, (0, 0))$. For $i > 0$, C_i is determined by C_{i-1}.

Let $(s_i^a, (x_i^a, y_i^a)) = C_i(a)$. We say that an agent a is in a cell (x, y) at time i if $(x_i^a, y_i^a) = (x, y)$, and that it has state s at time i if $s_i^a = s$. Now, the position of an agent a at time i is determined by applying the M function to the agent's state at time $i - 1$, and then updating its position from time $i - 1$ based on the result. Formally, (x_i^a, y_i^a) is determined by s_{i-1}^a and (x_{i-1}^a, y_{i-1}^a). For example, if $M(s_{i-1}^a) = \text{NORTH}$, then $(x_i^a, y_i^a) = (x_{i-1}^a, y_{i-1}^a + 1)$.

The state of an agent at time i is determined by applying the δ function to the agent's state at time $i - 1$, also taking as input whether or not the agent is alone or

at the origin at time i. Formally, define the variables $alone_i^a$ and $origin_i^a$ as

$$alone_i^a \coloneqq \bigwedge_{a' \neq a \in A} (x_i^{a'}, y_i^{a'}) \neq (x_i^a, y_i^a)$$

$$origin_i^a \coloneqq (x_i^a, y_i^a) = (0, 0).$$

We say that an agent a is alone at time i if $alone_i^a = \text{TRUE}$, and that it is at the origin at time i if $origin_i^a = \text{TRUE}$. Then s_i^a is determined by sampling the distribution returned by $\delta(s_{i-1}^a, alone_i^a, origin_i^a)$.

We say that an agent is in a phase at time i if the *phase* variable of s_i^a is that phase. If an agent switches from some phase to another in round i, then we say that the agent completes the former and enters the latter in round i.

2.3 Problem Statement

The goal is to locate a treasure in some cell distance D from the origin in as few expected rounds as possible. The distance of a cell from the origin is defined to be the minimum number of moves that an agent could make to get from the origin to the cell (i.e., Manhattan distance). The treasure is considered located once any agent is positioned in the same cell as the treasure.

2.4 Definitions

We need a few additional definitions in order to describe our algorithm. As we will see, RECTANGLE-SEARCH (see Chapter 3) involves the agents beginning by searching the cells nearby the origin, and progressively making their way to cells further from the origin. However, the algorithm is not quite centered around the origin. Instead, it is centered around a different cell, which we will call the *center*, as defined below.

Definition 2.1. The *center* of the grid is the cell $(1, -1)$.

This is due to the fact that the origin is a special cell since the agents can detect

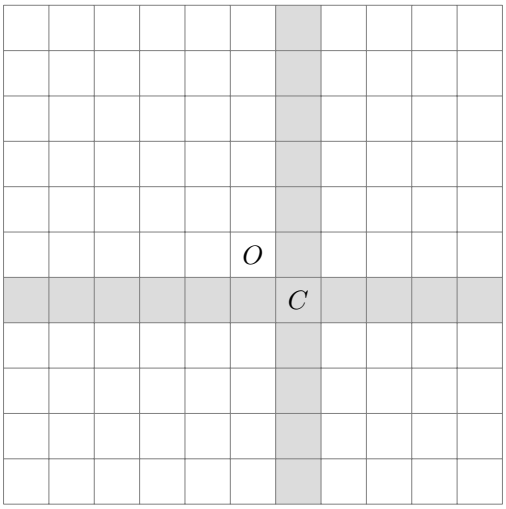


Figure 2-1: The cell marked O is the origin, and the cell marked C is the center. The cells located on an axis are shaded in gray.

whether they are in it. As we will see in Chapter 3, in order to maintain separation between different phases of RECTANGLE-SEARCH, the agents need the recognizable cell to be different from the cell around which the search is centered.

We also define the axes with respect to the center instead of the origin. Any cell that lies directly north of the center is on the *north axis*, and similarly for the east, south, and west axes. This is demonstrated in Figure 2-1.

The agents begin by searching all the cells exactly distance 1 from the center, and then searching all the cells exactly distance 2 from the center, and so on. To formalize this idea, we include the following definition.

Definition 2.2. *Level* d is the set of all cells exactly distance d from the center.

Note that because the center is two moves from the origin, the treasure is located at most $D + 2$ moves from the center. Thus, if the agents exhaustively search levels 0 through $D + 2$, they are guaranteed to locate the treasure.

Chapter 3

The Rectangle-Search Algorithm

In this chapter, we describe the algorithm Rectangle-Search. We prove that this algorithm locates the treasure in $O(D \cdot \log k + D^2/k)$ rounds in expectation when $k \geq 19$. Furthermore, we conjecture that it actually locates the treasure in $O(D \cdot \log \log k + D^2/k + \log k)$ rounds in expectation. For the remainder of this chapter we assume that $k \geq 19$.

Recall that initially all agents are located at the origin. At the start of the algorithm, each agent executes the *Separation Phase* (see Section 3.1). The purpose of the *Separation Phase* is for each agent to separate itself from the other agents. Once an agent has separated itself from the other agents, it is more powerful because it can make use of its ability to detect if it is alone.

Once an agent completes the *Separation Phase*, it immediately moves into the *Allocation Phase* (see Section 3.2). In this phase, the agents are assigned one of five roles, which determines how they behave in the following phase.

Immediately after having finished the *Allocation Phase*, an agent enters the *Search Phase* (see Section 3.3). Most of the exploration of the grid is done during the *Search Phase*. In this phase, the agents work in a carefully coordinated effort to exhaustively search the grid from the origin outwards.

We begin by describing the *Separation Phase* in Section 3.1. In this section we define functions δ-Separation and M-Separation, which determine how an agent updates its state and moves while in the *Separation Phase*. When proving facts about

the behavior of an agent A in the *Separation Phase*, we do not consider the impact that an agent in another phase could have on A (we will handle this in Section 3.4).

In Section 3.2 we present the *Allocation Phase*, and in doing so define functions δ-ALLOCATION and M-ALLOCATION. We prove statements about the behavior of an agent A in the *Allocation Phase*, again ignoring the impact agents in other phases could have on A until Section 3.4.

In Section 3.3, we introduce the final phase of RECTANGLE-SEARCH, the *Search Phase*. We define functions δ-SEARCH and M-SEARCH. Again, we do not consider the impact agents from other phases have on *Search Phase* agents until Section 3.4

Finally, in Section 3.4 we argue that none of the agents are able to impact the behavior of agents in other phases. Using this, we prove that RECTANGLE-SEARCH is guaranteed to locate the treasure eventually. We then analyze the expected time until some agent locates the treasure.

With these functions defined, we define the functions δ-RECTANGLE and M-RECTANGLE as follows. Recall that an agent is a tuple (S, s_0, δ, M). For an agent to execute RECTANGLE-SEARCH, it uses $\delta = \delta$-RECTANGLE and $M = M$-RECTANGLE. To define these functions we only need to use the *phase* variable of the state. Throughout this chapter, we introduce the variables of the state and their initial values as they are needed.

With this in mind, we present the pseudocode for RECTANGLE-SEARCH.

Each state $s \in S$ contains the following variable.

phase: member of {SEPARATION, ALLOCATION, SEARCH}, initially SEPARATION.

δ-RECTANGLE$(s, alone, origin)$
  if $s.phase =$ SEPARATION
    $s :=$ δ-SEPARATION$(s, alone, origin)$
  if $s.phase =$ ALLOCATION
    $s :=$ δ-ALLOCATION$(s, alone, origin)$
  if $s.phase =$ SEARCH
    $s :=$ δ-SEARCH$(s, alone, origin)$
  return s

```
M-RECTANGLE(s)
    if s.phase = SEPARATION
        return M-SEPARATION(s)
    if s.phase = ALLOCATION
        return M-ALLOCATION(s)
    if s.phase = SEARCH
        return M-SEARCH(s)
```

In Sections 3.1, 3.2, and 3.3, we describe each of the phases of RECTANGLE-SEARCH and define the functions used in the pseudocode above.

3.1 Separation Phase

In this section, we describe the *Separation Phase* of the algorithm. The purpose of this phase is for the agents to separate themselves.

Input Assumptions

1. Each agent enters the phase at the origin.
2. Agents enter the phase in the same even-numbered round.
3. Let k' be the total number of agents that begin the *Separation Phase*. Then $k' \geq 19$.

Guarantees

1. No two agents complete the phase in the same round.
2. Agents only complete the phase in even-numbered rounds.
3. Agents complete the phase at the origin.
4. The probability that at least 19 agents have completed the phase by round T approaches 1 as T goes to infinity.

3.1.1 The Algorithm

In this section, and in Sections 3.2 and 3.3, we begin by giving an informal description of the protocol for an agent in the phase, and then give formal pseudocode.

Informal Description

In rounds 1 and 2, the agent moves west. In round 3, it does not move. In round 4, it moves west with probability 1/2, and otherwise remains in the same position. It repeats this process of pausing on odd-numbered rounds and moving west with probability 1/2 only in even-numbered rounds until it finds itself alone in a cell. After detecting that it is alone, it continues to not move in odd-numbered rounds, but on even-numbered rounds it moves east. It repeats this process until it reaches the origin. Upon reaching the origin, the agent has completed the *Separation Phase*, and switches to the *Allocation Phase*.

Formal Description

Each state $s \in S$ contains the following variables. The variables that are not used in this phase are omitted here.

phase: member of {SEPARATION, ALLOCATION}, initially SEPARATION.
count: integer in {0, 1, 2}, initially 0
back: boolean, initially FALSE
even: boolean, initially FALSE
coin: members of {HEADS, TAILS}, initially HEADS

δ-SEPARATION($s, alone, origin$)
    $s.coin$:= result of fair coin toss
    if $s.count < 2$
        $s.count := s.count + 1$
    if *alone*
        $s.back$:= TRUE
    $s.even$:= not $s.even$
    if *origin* and $s.back$
        $s.phase$:= ALLOCATION
        Initialize other state variables for *Allocation Phase*
    return s

```
M-SEPARATION(s)
    if s.count < 2
        return WEST
    elseif s.even
        if s.back
            return EAST
        elseif s.coin = HEADS
            return WEST
    return NONE
```

3.1.2 Correctness

As we see in the Section 3.2, it is important for future phases that agents complete the *Separation Phase* in distinct rounds. It is also important that agents only complete the phase on even-numbered rounds at the origin. We formalize these ideas with the following theorem.

Theorem 3.1 (Guarantees 1, 2, and 3)**.** *Consider some agent A in a fixed execution α, and let T be the round in which A completes the Separation Phase in α. Then we have that*

1. *A is at the origin at time T.*
2. *T is even.*
3. *No other agent completes the Separation Phase in round T.*

Proof. The protocol dictates that A completes the phase when it reaches the origin, so part (1) is trivially true.

It is easy to see why T must be even. A moves west in rounds 1 and 2. After that, it alternates between pausing and moving, so the total number of rounds spent in the phase after that must be even, resulting in an even number of rounds overall. This proves part (2) of the theorem.

Part (3) is more interesting because it is not immediately obvious why it must be true. Consider some other agent A' executing this protocol. Suppose for the sake of

contradiction that A' also completes the *Separation Phase* in round T. Let $(-c, 0)$ be the furthest cell from the origin that A reached while executing the phase. We assume that A' made it at least as far from the origin as A while executing the phase. If this is not the case, then the symmetric argument proves this part.

For this to have been the furthest cell reached by A, it must have been the case that A was alone in one of the last two rounds in which it was in that cell. This is because after the first two initial moves west, A only moves every other round, so it spends two rounds in every cell. The protocol dictates that it switches to moving back east if it detects being alone in either of those two rounds. It is not obvious why this analysis is useful until we consider the interaction between *Separation Phase* agents and agents from other phases.

We know that A and A' are both at the origin at time T. Because they both deterministically move east every other round after detecting being alone, we know that they were both in the cell $(-1, 0)$ at times $T - 1$ and $T - 2$. Following this reasoning, both A and A' must have been in cell $(-c, 0)$ at times $T - 2c + 1$ and $T - 2c$. This contradicts the above statement that A was alone in one of the last two rounds it was in $(-c, 0)$, proving part (3) of the theorem.

□

In our final analysis of RECTANGLE-SEARCH in Section 3.4, we will show that the agents eventually locate the treasure with probability 1. To help prove this, we show that at least 19 agents eventually complete the phase with probability 1. First we use Lemma 3.2 to show this for a single agent, and then in Theorem 3.3 we use this to show it for 19 agents. Note that in Lemma 3.2 and Theorem 3.3, we analyze a *probabilistic* execution β, which is a tree consisting of all possible executions, as opposed to a fixed execution, which is a single branch in the tree.

Lemma 3.2. *Consider some agent A in a probabilistic execution β, and let T be the round in which A completes the Separation Phase in β. Then we have that for any*

$t \geq 8$,

$$P(T < t) \geq 1 - \left(1 - \frac{1}{2^{k'}}\right)^{\left\lfloor \frac{t-6}{4} \right\rfloor}$$

Proof. Before proving this lemma, we introduce an additional definition. We say that some round R is a *probabilistic round* if $R = 2r$ for some $R \geq 2$. In a probabilistic round, agents that have not yet become alone move west with probability $1/2$.

Let F_R be the event that A has become alone by probabilistic round R. We begin by upper bounding $P\left(\overline{F_R}\right)$. A will become alone in the first probabilistic round if it decides to move west and all other agents remain in place, or it decides to remain in place and all other agents move west. So we have

$$P(F_1) = 2 \cdot \frac{1}{2} \cdot \frac{1}{2^{k'-1}} = \frac{1}{2^{k'-1}}.$$

And then clearly

$$P\left(\overline{F_1}\right) = 1 - \frac{1}{2^{k'-1}}.$$

After probabilistic round R, let Y_w, Y_c, and Y_e be the number of agents in the cell west of A, in the cell which A is in (not including A), and in the cell east of A, respectively. For A to not have become alone by probabilistic round R, we must first have that A did not become alone in any previous round. Furthermore, if A chooses to move west in probabilistic round R, then at least one other agent which had been sharing A's cell must move west or at least one agent which had been in the cell west of A must remain in place, and similarly if A chooses to remain in place. So we have

$$\begin{aligned}
P\left(\overline{F_R}\right) &= P\left(\overline{F_1}\right) \cdot \ldots \cdot P\left(\overline{F_{R-1}}\right) \cdot \left(1 - \left(\frac{1}{2} \cdot \frac{1}{2^{Y_w+Y_c}} + \frac{1}{2} \cdot \frac{1}{2^{Y_c+Y_e}}\right)\right) \\
&\leq P\left(\overline{F_1}\right) \cdot \ldots \cdot P\left(\overline{F_{R-1}}\right) \cdot \left(1 - \frac{1}{2^{k'-1}}\right) \\
&\leq \left(1 - \frac{1}{2^{k'-1}}\right)^R,
\end{aligned}$$

leading to

$$P(F_R) \geq 1 - \left(1 - \frac{1}{2^{k'-1}}\right)^R.$$

Recall that all agents move west in rounds 1 and 2, and then pause every other round until becoming alone. So an agent that becomes alone in probabilistic round R becomes alone in round $2R + 2$. In that round that agent is at most $R + 2$ moves from the origin. Since it then moves east every other round until reaching the origin, we know that it completes the phase by round $4R + 6$. This yields the relationship $T \leq 4R + 6$. Rearranging, we have have $R \geq \lfloor (T - 6)/4 \rfloor$. So, if an agent spends t rounds in the phase, then it spent at least $\lfloor (t-6)/4 \rfloor$ rounds probabilistically deciding whether to move west. This leads to

$$P(T < t) \geq 1 - \left(1 - \frac{1}{2^{k'-1}}\right)^{\lfloor \frac{t-6}{4} \rfloor},$$

as desired. □

Now, with Lemma 3.2 in mind, we can prove Theorem 3.3, which lower bounds the probability that at least 19 agents complete the phase by the end of some round.

Theorem 3.3 (Guarantee 4)**.** *Consider a probabilistic execution β, let $A_1, \ldots A_{19}$ be random variables representing the 1^{st} through 19^{th} agents to complete the Separation Phase in β, and let T be a random variable representing the round in which A_{19} does so. Then we have that for any $t \geq 8$,*

$$P(T < t) \geq \left(1 - \left(1 - \frac{1}{2^{k'}}\right)^{\lfloor \frac{t-6}{4} \rfloor}\right)^{19}$$

Proof. Let F_i be the event that agent A_i completes the phase by round T. Then by

Lemma 3.2, we have

$$P(F_1) \geq 1 - \left(1 - \frac{1}{2^{k'}}\right)^{\left\lfloor \frac{T-6}{4} \right\rfloor}.$$

Now, we want to compute the probability that two agents complete the phase by round T, namely $P(F_1 \wedge F_2)$. Clearly F_1 and F_2 are not independent events. However, we claim that the occurrance of F_1 only makes the occurrance of F_2 more likely. This is because once one agent completes the phase, all other agents have fewer agents that they might be sharing a cell with, and are therefore more likely to become alone. So, we have

$$\begin{aligned} P(F_1 \wedge F_2) &\geq P(F_1) \cdot P(F_2) \\ &\geq \left(1 - \left(1 - \frac{1}{2^{k'}}\right)^{\left\lfloor \frac{T-6}{4} \right\rfloor}\right)^2 \end{aligned}$$

Applying this same reasoning to the next 17 agents, we have

$$\begin{aligned} P(F_1 \wedge \ldots \wedge F_{19}) &\geq P(F_1) \cdot \ldots \cdot P(F_{19}) \\ &\geq \left(1 - \left(1 - \frac{1}{2^{k'}}\right)^{\left\lfloor \frac{T-6}{4} \right\rfloor}\right)^{19}, \end{aligned}$$

as desired. □

The following lemma outlines the region in which an agent in the *Separation Phase* could be located. This is useful for us when we combine the *Separation Phase* with other phases and need to show that agents in different phases do not interact with each other. The region is displayed in Figure 3-1.

Lemma 3.4. *Consider some agent A in the Separation Phase. Then A's current cell is $(-c, 0)$ for some $c \geq 0$.*

Proof. A begins the *Separation Phase* at the origin, and immediately moves west to the cell $(-1, 0)$. In all future rounds until it is alone, it only moves west, so its position

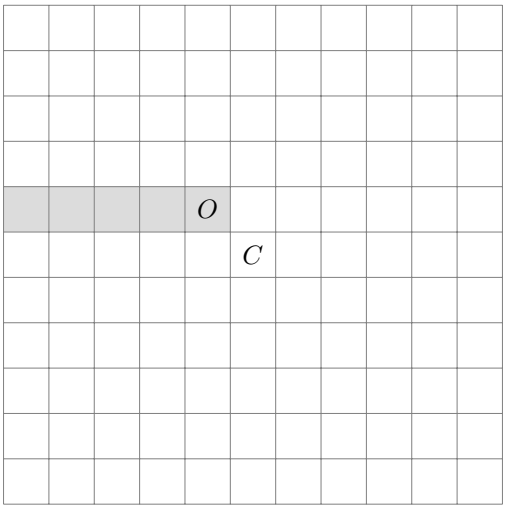


Figure 3-1: The gray region indicates the cells in which an agent from the *Separation Phase* could be located. Note that the region extends infinitely far to the west. The cell marked O is the origin, and the cell marked C is the center.

must be of the form $(-c, 0)$ for some $c > 0$.

After becoming alone the agent switches to only making moves to the east. However, since it leaves the *Separation Phase* as soon as it reaches the origin, it can never be in the *Separation Phase* in any cell east of the origin. □

3.1.3 Analysis

The *Separation Phase* is the only randomized phase in RECTANGLE-SEARCH. So, we need to upper bound the expected number of rounds until the agents complete this phase. However, since some agents are moving on to other phases while other agents are still in the *Separation Phase*, it is not enough to compute the number of rounds until all agents complete the phase. Instead, we want to upper bound the number of rounds until a given number of agents have completed the phase.

Let T_i be a random variable representing the round in which the i^{th} agent completes the *Separation Phase*. Our goal in this section is to upper bound $E[T_i]$. To this end, we define the *Separation completion function* $f_k(i) : \{1, \ldots, k\} \to \mathbb{N}$ to be the expected number of rounds until i agents have completed the *Separation Phase*.

Formally, we define $f_k(i) = E[T_i]$. We prove that $f_k(i) = O(i \cdot \log k)$. Furthermore, we conjecture that after the first agent completes the phase, the other agents complete it in even fewer rounds. Specifically, we conjecture that $f_k(i) = O(\log k + i \cdot \log \log k)$.

Before beginning with those proofs, we define a few more random variables that we will use throughout this section. Recall that we defined a *probabilistic round* to be any round T such that $T = 2r$ for some $r \geq 2$. Let R_i be number of probabilistic rounds until the i^{th} agent becomes alone. Note that it is not necessarily the case that the i^{th} agent that becomes alone is the i^{th} agent to complete the phase. If an agent becomes alone far from the origin, and another agent later becomes alone closer to the origin, the latter may complete the phase first.

Recall in the proof of Theorem 3.1 that we reasoned about the relationship between the number of probabilistic rounds and the number of rounds an agent spends in the phase. The i^{th} agent becomes alone in round $2R_i + 2$, and in that round that agent is at most $R_i + 2$ moves from the origin. It then moves east every other round until reaching the origin, so it completes the phase by round $4R_i+6$, leading to $T_i \leq 4R_i+6$.

Let cell C_T be a random variable representing the closest cell to the origin that contains agents that have not yet become alone after T probabilistic rounds, and let Y_T be a random variable representing the number of such agents in that cell after that round. We say that a probabilistic round T is *counted* if $Y_T = 1$. Let Z_i be the i^{th} probabilistic round that is counted. Through the following lemmas, we upper bound $E[Z_i]$, and then use that to upper bound $E[T_i]$. We begin by establishing a simple property of C_T which will be useful in later proofs.

Lemma 3.5. *Let α be a fixed execution, and let x_T be the x-coordinate of cell C_T in α. Then for any $T > 1$, $x_T \leq x_{T-1}$.*

Proof. Let S_T be the set of agents that have not yet been alone by time T. By definition, at time $T - 1$ all agents in S_{T-1} have x coordinate at most x_{T-1}. Note that S_T is a subset of S_{T-1}. Consider some agent A in S_{T-1}. If A is also in S_T, then it

did not become alone in round T, and therefore it either moved west or remained in the same position. Thus, its x coordinate is still at most x_{T-1}, proving the claim. $\square$

We use the next two lemmas to upper bound the expected number of probabilistic rounds between counted probabilistic rounds. To do so, we begin by using Lemma 3.6 to provide a lower bound on the probability that certain specific rounds are counted. Then, in Lemma 3.7, we use that probability to upper bound the expected number of rounds until a counted probabilistic round occurs.

Lemma 3.6. *Consider some fixed finite execution α that contains exactly T probabilistic rounds for some $T \geq 0$, where α ends on a probabilistic round if $T > 0$. Furthermore, in α, there is at least one agent that does not become alone. Let β be a probabilistic extension of α, resulting from extending α another $\lceil \log Y_T \rceil + 1$ probabilistic rounds. Then the probability that β contains a counted round after the prefix α is at least $1/(4\sqrt{e})$.*

Proof. Consider a modified version of the algorithm in which after becoming alone, agents continue tossing coins and remain in their cells as *ghosts*, meaning that other agents do not detect their presence when sharing their cell. Let γ be a fixed finite execution of this modified algorithm that contains exactly T probabilistic rounds, and let ζ be a probabilistic extension of γ, resulting from extending γ another $\lceil \log Y_T \rceil + 1$ probabilistic rounds. Let S be the set of agents in cell C_T at the end of probabilistic round T. Let F be the event that ζ contains a round in which exactly one agent from S is in cell C_T after the prefix γ. Then F is guaranteed to occur if

1. One agent from S remains in place in every probabilistic round for $\lceil \log Y_T \rceil + 1$ probabilistic rounds.

2. The $Y_T - 1$ other agents from S each have at least one probabilistic round in the next $\lceil \log Y_T \rceil + 1$ probabilistic rounds where they move west.

So then we can lower bound the probability of F occurring by

$$\begin{aligned}
P(F) &\geq Y_T \cdot \left(\frac{1}{2^{\lceil \log Y_T \rceil + 1}}\right) \cdot \left(1 - \frac{1}{2^{\lceil \log Y_T \rceil + 1}}\right)^{Y_T - 1} \\
&\geq Y_T \cdot \left(\frac{1}{2^{\log Y_T + 2}}\right) \cdot \left(1 - \frac{1}{2^{\log Y_T + 1}}\right)^{Y_T - 1} \\
&= Y_T \cdot \left(\frac{1}{4Y_T}\right) \cdot \left(1 - \frac{1}{2Y_T}\right)^{Y_T - 1} \\
&= \frac{1}{4} \cdot \left(1 - \frac{1}{2Y_T}\right)^{Y_T - 1} \\
&\geq \frac{1}{4\sqrt{e}}.
\end{aligned}$$

Consider any execution γ of the modified algorithm. The corresponding execution of the unmodified algorithm is the execution of the same number of rounds in which the results of the coin flips are the same as those in γ (for as long as coin flips are required in the unmodified algorithm).

Let G be the event that β contains a counted round after the prefix α. Consider an execution ζ in which F occurs. Then in the corresponding execution of the unmodified algorithm, G occurs. As a result, we know $P(G) \geq P(F)$, proving the claim.

□

With this in mind, we now use the following lemma to establish the expected number of rounds until the next counted probabilistic round.

Lemma 3.7. *Consider some fixed finite execution α that contains exactly T probabilistic rounds for some $T \geq 0$, where α ends on a probabilistic round if $T > 0$. Furthermore, in α, there is at least one agent that does not become alone. Let β be a probabilistic extension of α, resulting from extending α until it ends on a counted round. Then the expected number of rounds required to extend α is at most $4\sqrt{e} \cdot (\lceil \log k \rceil + 1)$.*

Proof. By Lemma 3.6, with probability at least $1/(4\sqrt{e})$, an extension of α of $\lceil \log Y_T \rceil +$

1 probabilistic rounds contains a counted round. Since we know $Y_T \leq k$, the probability that an extension of α of $\lceil \log k \rceil + 1$ probabilistic rounds contains a counted round is at least $1/(4\sqrt{e})$.

Say that an extension of α of $\lceil \log k \rceil + 1$ probabilistic rounds does not contain a counted round. By the same reasoning, an extension of another $\lceil \log k \rceil + 1$ probabilistic rounds also has probability at least $1/4\sqrt{e}$ of containing a counted round. If that extension still does not contain a counted round, we can repeat this process.

Let W be a random variable representing the number of times we must repeat this process before a round is counted. Then we have that a round is counted once we have extended α by $W \cdot (\lceil \log k \rceil + 1)$ probabilistic rounds. Since each repetition of this process has probability at least $1/(4\sqrt{e})$ of success, W is a geometric random variable with parameter $p \geq 1/(4\sqrt{e})$, so $E[W] \leq 4\sqrt{e}$. So then we have that we expect to need to extend α by at most $4\sqrt{e} \cdot (\lceil \log k \rceil + 1)$ probabilistic rounds, as desired.

□

With these lemmas in mind, we can now upper bound the number of rounds until i agents have completed the *Separation Phase* in expectation. We begin by proving in Lemma 3.8 that some agent completes the phase within $O(\log k)$ rounds in expectation. Then, in Lemma 3.9, we prove that i agents complete the phase in $O(i \cdot \log k)$ rounds in expectation.

Lemma 3.8. *Some agent completes the Separation Phase within $O(\log k)$ rounds in expectation. Formally, $E[T_1] = O(\log k)$.*

Proof. Setting $T = 0$ and applying Lemma 3.7, we have

$$E[Z_1] \leq 4\sqrt{e} \cdot (\lceil \log k \rceil + 1)\,.$$

We argue that $R_1 \leq Z_1$. Combining this with the fact that $T_1 \leq 4R_1 + 6$, which we

argued earlier, proves the claim.

We know that in probabilistic round Z_1 we have $Y_{Z_1} = 1$. Let A be a random variable representing that single agent. Then we consider two cases.

1. A is alone in probabilistic round Z_1.

   It directly follows that $R_1 \leq Z_1$.

2. A is sharing its cell with some agent A' in probabilistic round Z_1.

   By the definition of a counted round, A is the only agent in the cell that had not been alone in any previous round. So, A' must have been alone in some previous round. Since A' was alone before probabilistic round Z_1, we know $R_1 < Z_1$.

□

Lemma 3.9. *The Separation completion function satisfies* $f_k(i) = O(i \cdot \log k)$.

Proof. For any $i > 1$, consider the probabilistic round Z_{i-1}. By Lemma 3.7, we expect another counted round to occur within $4\sqrt{e} \cdot (\lceil \log k \rceil + 1)$ probabilistic rounds of probabilistic round Z_{i-1}. So then we have

$$E[Z_i | Z_{i-1}] \leq Z_{i-1} + 4\sqrt{e} \cdot (\lceil \log k \rceil + 1) .$$

Now, with this in mind, using iterated expectation, this means that

$$\begin{aligned} E[Z_i] &= E[E[Z_i | Z_{i-1}]] \\ &\leq E[Z_{i-1}] + 4\sqrt{e} \, (\cdot \lceil \log k \rceil + 1) . \end{aligned}$$

Recursively applying yields

$$E[Z_i] \leq E[Z_1] + (i-1) \cdot \left(4\sqrt{e} \cdot (\lceil \log k \rceil + 1)\right) .$$

By the proof of Lemma 3.8, we have $E[Z_1] \leq 4\sqrt{e} \cdot (\lceil \log k \rceil + 1)$. So then we have

$$E[Z_i] \leq i \cdot \left(4\sqrt{e} \cdot (\lceil \log k \rceil + 1)\right).$$

It remains to show that $R_i \leq Z_i$.

Let x_{Z_i} be the x-coordinate the cell C_{Z_i}. We prove using induction that by the end of probabilistic round Z_i, at least i distinct agents have become alone and then afterwards reached a cell (x, y) such that $x \geq x_{Z_i}$. This directly implies that $R_i \leq Z_i$. For the base case, we showed in the proof of Lemma 3.8 that in round Z_1 there is an agent that has become alone in cell C_{Z_1}.

Now, assume for induction that by the end of round Z_{i-1}, there have been at least $i - 1$ distinct agents that have become alone and reached a cell (x, y) such that $x \geq x_{Z_{i-1}}$. Let S_{i-1} be the set of those $i - 1$ agents. We prove that after Z_i probabilistic rounds, there is an agent in cell C_{Z_i} which is not in S_{i-1} and has already become alone.

We proceed with a proof very similar to that of Lemma 3.8. We know that in probabilistic round Z_i we have $Y_{Z_i} = 1$. Let A be a random variable representing that single agent. We consider two cases.

1. A is alone in probabilistic round Z_i.

   Since A becomes alone for the first time in probabilistic round Z_i, we know that it is not in S_{i-1}.

2. A is sharing its cell with some other agent A' in probabilistic round Z_i.

   Since Z_i is counted, we know that A' become alone in some probabilistic round before Z_i. But since agents only move east after becoming alone, we know that that to be in C_{Z_i} in round Z_i, A' must have become alone in a cell west of C_{Z_i}. But then we know that in the rounds between when A' became alone and Z_i,

A' was west of C_{Z_i}, and therefore by Lemma 3.5 also west of $C_{Z_{i-1}}$. Therefore, we know that A' is not in S_{i-1}.

□

We believe that our bound on the number of rounds for the first agent to complete the *Separation Phase* is tight. That is, it seems to be the case that $E[T_1] = \Theta(\log k)$. However, for any $i > 1$, it does not seem to be the case that $E[T_i] = \Theta(i \cdot \log k)$. In fact, we believe that after the first agent completes the phase, another agent completes the phase roughly every $O(\log \log k)$ rounds. The following conjecture states this claim and provides some intuition behind it.

Conjecture 3.10. *The Separation completion function satisfies* $f_k(i) = O(\log k + i \cdot \log \log k)$.

Intuition. We claim that for any $T \geq \log k$, $E[Y_T] = O(\log k)$. We provide our intuition for why this claim is true, but do not have a formal proof. The formal proof of this claim is the only missing piece of the proof of this conjecture.

We begin by arguing that $E[Y_{\log k}] = O(\log k)$. With high probability, in the first $\log k$ rounds at least one agent moved west no more than once. Since the expected number of agents that did not move west at all is 1, and the expected number that moved west only once is $\log k$, we know that $E[Y_{\log k}] = O(\log k)$. Since the agents are spreading out over time, intuitively we believe that $E[Y_T]$ decreases as T increases.

By Jensen's inequality, we know that $E[\log Y_T] \leq \log E[Y_T]$. Recall in the proof of Lemma 3.6, we used the fact that $\log Y_T \leq \log k$. If we replace this fact with

$$\begin{aligned} E[\log Y_T] &\leq \log E[Y_T] \\ &= \log\left(O(\log k)\right) \\ &= O(\log \log k), \end{aligned}$$

then the same logic in the proof leads to the fact that for any $T \geq \log k$, the next counted probabilistic round occurs within $O(\log \log k)$ rounds in expectation.

Using this as we did in the proof of Lemma 3.9, this leads to

$$E[Z_i | Z_{i-1}] \leq Z_{i-1} + O(\log \log k).$$

Recursively applying yields

$$E[Z_i] = O(\log k + i \cdot \log \log k).$$

The reasoning that $R_i \leq Z_i$ is still valid, so this leads to

$$E[T_i] = O(\log k + i \cdot \log \log k).$$

3.2 Allocation Phase

In this section we describe the *Allocation Phase*. Unlike the *Separation Phase*, the *Allocation Phase* is deterministic. The purpose of the *Allocation Phase* is to assign each agent a role and to ensure that the agents enter the *Search Phase* in the appropriate time and place. Note that the input assumptions in this phase are guarantees provided by the *Separation Phase*. Similarly, we will see that the input assumptions of the *Search Phase* are a subset of the Guarantees of this phase. Guarantees 4 and 9 of the *Allocation Phase* are not input assumptions of the *Search Phase*, but will be useful in the overall analysis of RECTANGLE-SEARCH.

Input Assumptions

1. No two agents enter the phase in the same round.

2. Agents only enter the phase in even-numbered rounds.

3. Agents enter the phase at the origin.

4. Let k' be the total number of agents that ever enter the *Allocation Phase*. Then $k' \geq 19$.

Guarantees

1. Every agent that completes the phase is assigned one of five roles: North Guide, East Guide, South Guide, West Guide, or Explorer.

2. All North Guides complete the phase four cells north of the center, East Guides four cells east of the center, South Guides four cells south of the center, and West Guides and Explorers four cells west of the center.

3. The phase produces at least two agents of each role.

4. Once all the agents that will ever complete the phase have done so, the number of Explorers is at least $\lfloor (k' - 9)/5 \rfloor$ and at most $\lfloor (k' - 1)/5 \rfloor$.

5. Upon completing the phase, each agent knows whether it is the first of its role to complete the phase.

6. The second agent of any given role completes the phase exactly two rounds after the first agent of the role, and there are at least 8 rounds between when any two other agents of that role complete the phase.

7. A West Guide always completes the phase in the round after an Explorer completes the phase.

8. Every Explorer completes the phase in an even-numbered round.

9. *Allocation Phase* agents search levels 0 through 3 completely.

10. Before the first Explorer completes the phase, a North Guide, East Guide, and South Guide have completed the phase.

3.2.1 The Algorithm

Each agent in this phase executes either *Part A* or *Part B*. We will see that *Part A* produces exactly one agent of each role and that the agents executing *Part B* are equally split among the roles. Both parts are similar, in that they involve each agent moving in a carefully designed path around the center, along the way using its ability to detect loneliness to determine its role. These paths are designed to have minimal overlap to simplify some of our proofs, but there are many other paths that would also produce a correct algorithm.

As we will see, the first agent of each role to complete the phase is the first agent of that role that executes *Part B*. These agents know that they are the first of their role to complete the phase because they will have shared a cell with the *Part A* agent of that role before completing the phase.

Informal Description

An agent that has just entered the *Allocation Phase* first senses whether it is alone at the origin. If so, it executes *Part A* of the *Allocation Phase*. Otherwise, it executes *Part B*. We will see that some agents executing *Part A* will return to the origin, so that once they do so, all subsequent agents to enter the phase will execute *Part B*.

Part A: An agent executing *Part A* of the *Allocation Phase* follows the path outlined in Figure 3-2, by moving to the next cell in the path each round. If the agent reaches the cell marked N and detects that it is alone, then it stops following the path and begins executing the North Guide protocol (to be described shortly). Similarly, if it is alone in the cell marked E, S, W, or X, it stops following the path and begins executing the East Guide, South Guide, West Guide, or Explorer protocol, respectively. Otherwise, the agent continues following the path back to the origin, where it remains for the rest of the algorithm. If the agent returns to the origin, it never completes

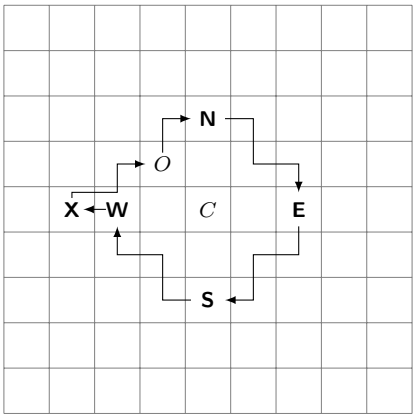


Figure 3-2: The line represents the path followed by an agent in *Part A* of the *Allocation Phase*. Agents begin at the origin, marked O. The cell marked C is the center. If an agent reaches one of the cells marked N, E, S, W, or X and detects that it is alone, it stops following the path and begins executing the appropriate role protocol.

the *Allocation Phase*.

Now we describe the protocol for an agent that has been assigned one of the roles. An agent assigned any role begins by waiting in its current cell until it shares it with another agent in two consecutive rounds. As we will see, this guarantees that the agent remains in its cell until sharing it with some *Part B* agent of the same role. Afterwards, if the role is Explorer, then the agent does not move for one round and then moves west once, after which it has completed the *Allocation Phase*. If the role is a Guide, then after detecting it is not alone in two consecutive rounds, the agent waits to be alone and then does not move for two rounds and then moves twice in its cardinal direction. After that, it has completed the *Allocation Phase*. After completing the *Allocation Phase*, the agent immediately switches to the *Search Phase*.

Part B: An agent executing *Part B* of the *Allocation Phase* is very similar to an agent executing *Part A*. The agent follows the path outlined in Figure 3-3. If it finds itself alone in the cell marked N, E, S, or W, it stops and begins executing the protocol for a North Guide, East Guide, South Guide, or West Guide, respectively. Otherwise,

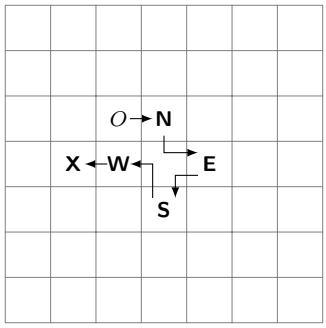


Figure 3-3: The line represents the path followed by an agent in *Part B* of the *Allocation Phase*. Agents begin at the origin, marked O. If an agent reaches one of the cells marked N, E, S, or W and detects that it is alone, it stops following the path and begins executing the appropriate role protocol. Otherwise it moves to the cell marked X and becomes an Explorer.

it reaches the cell marked X and becomes an Explorer.

If the agent is assigned the role of Explorer, then it moves west twice, after which it has completed the *Allocation Phase*. If the role is West Guide, then the agent waits until it shares its cell in a round. Then it does not move for one round, then moves west in the following three rounds, after which is has completed the *Allocation Phase*.

If the role is North, East, or South Guide, then the agent waits until it shares its cell in a specified number of (not necessarily consecutive) rounds. This number is 4 for North Guides, 3 for East Guides, and 2 for South Guides. These values are chosen to guarantee that every fifth agent is assigned the same role. Then it moves in its cardinal direction for one round, does not move for one round, and then moves in its cardinal direction for two rounds. After that, it has completed the *Allocation Phase*. After completing the *Allocation Phase*, the agent immediately switches to the *Search Phase*.

If the agent detects that it is not alone in the round of its first move while executing a role protocol, it concludes that it is the first agent of its role to complete the *Allocation Phase*. Otherwise, it concludes that some other other agent of its role was the first to complete the phase.

Formal Description

Note that in order to define the state transition function δ-ALLOCATION for this section, we use a helper function δ-NESWX-ALLOCATION.

Each state $s \in S$ contains the following variables. The variables that are not used in this phase are omitted here.

phase: member of {ALLOCATION, SEARCH}, initially ALLOCATION
part: member of {A, B}, initially B
role: member of {NONE, NORTH, EAST, SOUTH, WEST, EXPLORER}, initially NONE
moves: array of members of {NORTH, EAST, SOUTH, WEST},
initially [EAST, SOUTH, EAST, WEST, SOUTH, NORTH, WEST, WEST]
count: integer in $\{0, \ldots 18\}$, initially 0
rounds: integer in $\{0, \ldots, 5\}$, initially 0
shared: integer in $\{0, \ldots, 5\}$, initially 0
$first$: boolean, initially FALSE

δ-ALLOCATION($s, alone, origin$)
if $s.role =$ NONE
if *origin* and *alone* and $s.count = 0$
$s.part \coloneqq$ A
$s.moves \coloneqq$ [NORTH, EAST, EAST, SOUTH, EAST, SOUTH,
SOUTH, WEST, SOUTH, WEST, WEST, NORTH,
WEST, NORTH, WEST, EAST, NORTH, EAST]
if *alone*
if ($s.part =$ A and $s.count = 1$) or ($s.part =$ B and $s.count = 0$)
$s.role \coloneqq$ NORTH
if ($s.part =$ A and $s.count = 5$) or ($s.part =$ B and $s.count = 2$)
$s.role \coloneqq$ EAST
if ($s.part =$ A and $s.count = 9$) or ($s.part =$ B and $s.count = 4$)
$s.role \coloneqq$ SOUTH
if ($s.part =$ A and $s.count = 13$) or ($s.part =$ B and $s.count = 6$)
$s.role \coloneqq$ WEST
if ($s.part =$ A and $s.count = 14$)
$s.role \coloneqq$ EXPLORER
if ($s.part =$ B and $s.count = 7$)
$s.role \coloneqq$ EXPLORER
if not *origin* or $s.count = s.moves.length - 1$
$s.count \coloneqq s.count + 1$
if $s.role \neq$ NONE
$s \coloneqq \delta$-NESWX-ALLOCATION($s, alone, origin$)
return s

```
δ-NESWX-Allocation(s, alone, origin)
    if s.part = A
        if alone
            s.shared := 0
        elseif s.shared < 2
            s.shared := s.shared + 1
        if s.rounds > 0 or s.shared = 2
            s.rounds := s.rounds + 1
    elseif s.part = B
        if not alone
            if s.shared < 5
                s.shared := s.shared + 1
            if s.rounds = 1 or 2
                s.first := TRUE
        if s.rounds > 0 or
          (s.role = NORTH and s.shared ≥ 4) or
          (s.role = EAST and s.shared ≥ 3) or
          (s.role = SOUTH and s.shared ≥ 2) or
          (s.role = WEST and s.shared ≥ 1) or
          (s.role = EXPLORER)
            s.rounds := s.rounds + 1
    if s.rounds = 5 or (s.rounds = 3 and s.role = EXPLORER)
        s.phase := SEARCH
        Initialize other state variables for Search Phase
    return s


M-Allocation(s)
    if s.role = NONE and s.count < s.moves.length
        return s.moves[s.count]
    else
        if s.part = A
            if s.role = EXPLORER and s.count = 2
                return WEST
            elseif s.count ≥ 3
                return s.role
        elseif s.part = B
            if s.role = EXPLORER and s.rounds ≥ 1
                return WEST
            elseif s.role = WEST and s.rounds ≥ 2
                return WEST
            elseif (s.role = NORTH or EAST or SOUTH) and (s.rounds = 1 or ≥ 3)
                return s.role
    return NONE
```

3.2.2 Correctness

In this section, we consider some fixed execution α of the algorithm. We prove that all of the guarantees are satisfied in α, provided that the input assumptions are met. We begin with a simple theorem which proves that all agents are assigned a role before completing the phase.

Theorem 3.11 (Guarantee 1)**.** *Consider some agent A that completes the Allocation Phase, and let T be the round in which it does so. Then by round T, A has been assigned some role.*

Proof. First consider the case where A executes *Part A*. If A reaches the end of the path in Figure 3-2, then it never completes the phase. If it does not reach the end of the path, then it must have left the path early to execute some role protocol, and therefore is assigned some role before completing the phase.

Now consider the case where A executes *Part B*. If it reaches the end of the path in Figure 3-3, then it becomes an Explorer. Otherwise, it must have left the path early to execute some role protocol, and so it has been assigned some other role before completing the phase. □

Next, we present another simple theorem which establishes the cells in which agents complete the phase.

Theorem 3.12 (Guarantee 2)**.** *Let N be a North Guide, E an East Guide, S a South Guide, W a West Guide, and X an Explorer. Then*

1. *N completes the phase four moves north of the center.*
2. *E completes the phase four moves east of the center.*
3. *S completes the phase four moves south of the center.*
4. *W and X complete the phase four moves west of the center.*

Proof. Say that N is a *Part A* agent. Then N becomes a North Guide in the cell two moves north of the center. The *Part A* North Guide protocol requires exactly two moves north, so N will complete the phase four moves north of the center, as desired.

Now say that N is a *Part B* agent. Then it becomes a North Guide in the cell one move north of the center. The *Part B* North Guide protocol requires exactly three moves north, so it will complete the phase four moves north of the center.

A similar analysis proves the theorem for the other roles. □

We show in Lemma 3.13 that agents who execute *Part B* of the *Allocation Phase* are evenly distributed among the 5 roles. Then, in Lemma 3.14, we show that *Part A* produces exactly one agent of each role. Furthermore, we show that only a small number of agents ever execute *Part A* at all. Recall that an agent that reaches the end of the path in Figure 3-2 is at the origin and then remains there, so once some agent does that then all subsequent agents to enter the phase execute *Part B*.

Consider 5 consecutive agents who enter the phase and begin executing *Part B*. The protocol guarantees that each of those 5 agents are assigned different roles. This ensures that there are enough agents in each role. To get an intuition for why this is the case, we show a sample execution of *Part B* of the *Allocation Phase* in Figure 3-4. In this sample execution, a new agent enters the phase every other round. Subfigures show the positions of the agents at the end of consecutive rounds. Once an agent is assigned a role, it is shown in red. With this in mind, we present the following lemma.

Lemma 3.13. *Let B_i be the i^{th} agent to begin executing Part B of the Allocation Phase. Then we have that*

1. *If $i \pmod 5 \equiv 1$, then B_i becomes a North Guide.*
2. *If $i \pmod 5 \equiv 2$, then B_i becomes an East Guide.*
3. *If $i \pmod 5 \equiv 3$, then B_i becomes a South Guide.*

Figure 3-4: This figure depicts a sample execution of *Part B* of the *Allocation Phase*, in which a new agent enters the phase every other round. The subfigures show the agents at the end of consecutive rounds. The agent B_i is the i^{th} agent to begin executing *Part B*. Once an agent is assigned a role it is displayed in red. The origin is shown in gray.

4. *If* $i \pmod 5 \equiv 4$*, then* B_i *becomes a West Guide.*

5. *If* $i \pmod 5 \equiv 0$*, then* B_i *becomes an Explorer.*

Proof. First, note that no *Part A* agent ever enters one of the cells marked N, E, S, or W in Figure 3-3. So then we know that B_1 is the first agent to ever reach the cell marked N in Figure 3-3. It is therefore alone when it arrives there and becomes a North Guide.

Now let $i \equiv 1 \pmod 5$ for some $i > 1$, and assume that B_i becomes a North Guide. We will induct on i to prove the claim. As soon as it becomes a North Guide, it waits in the cell marked N in Figure 3-3 until it has shared its cell in 4 rounds. Since only *Part B* agents ever enter that cell, and no other agent pauses in that cell while it is occupied by B_i, we know that four separate *Part B* agents need to pass by that cell before B_i moves. Since every *Part B* agent passes through that cell, we know that B_{i+1}, B_{i+2}, B_{i+3}, and B_{i+4} all pass that cell while B_i is located there.

In the round after B_i shares its cell with B_{i+4}, B_i moves north. Thus, when B_{i+5} enters the cell marked N it is unoccupied, so B_{i+5} becomes a North Guide. This proves part (1) of the claim. A similar analysis proves the other parts of the claim.

□

As we have mentioned before, *Part A* produces exactly one agent of each role. In Lemma 3.12, we formally prove this. This allows us to refer to *the Part A* agent of some role. We will see in future proofs that this will simplify our logic. This lemma also bounds the number of agents that execute *Part A* and are never assigned a role.

Lemma 3.14. *Let* A_i *be the* i^{th} *agent to begin executing Part A of the Allocation Phase, and let* A_j *be the final agent to begin executing Part A. Then the following statements are true.*

1. $A_1, \ldots, A_5$ *are assigned distinct roles.*

2. *Any agent assigned a role remains in its respective cell in Figure 3-2 until sharing it with some Part B agent.*

3. *For any i, $5 < i \leq j$, A_i is never assigned a role.*

4. $6 \leq j \leq 14$.

Proof. An agent entering the *Allocation Phase* begins to execute *Part A* if it is alone in its first round in the phase. Every agent moves away from the origin as soon as it enters the phase, so if an agent is not alone at the origin, it must be sharing its cell with a *Part A* agent that has reached the end of the path in Figure 3-2.

We claim that A_6 is the first agent to execute *Part A* and reach the end of the path. A_1 is alone in the cell marked N and becomes a North Guide. It remains in that cell until it shares it in two consecutive rounds. While the cell is occupied, no other *Part A* agent remains in the cell in two consecutive rounds. A *Part A* agent could never occupy the cell in the round directly after another *Part A* agent, because by Input Assumption 2, agents only enter the phase in even-numbered rounds. So then as long as only *Part A* agents are in the phase, A_1 remains in that cell.

Similar arguments show that A_2, A_3, A_4, and A_5 reach and remain in the cells E, S, W, and X, respectively, for as long as only *Part A* agents are in the phase. So, when A_6 enters the phase, it moves past each of those cells and ultimately returns to the origin, proving that $j \geq 6$.

A_6 returns to the origin 18 rounds after it leaves the origin. During that time, since agents enter the *Allocation Phase* at most every other round, at most 8 other agents can begin *Part A*. Once A_6 returns to the origin, all agents entering the phase begin executing *Part B*, giving us $j \leq 6 + 8 = 14$.

It remains to show that those *Part A* agents that enter the phase after A_6 are not assigned any role. We have already argued that no *Part A* agent assigned a role leaves its corresponding cell in the path in Figure 3-2 until it shares that cell with

some *Part B* agent. So, we show no *Part A* agent ever enters one of those cells after a *Part B* agent has, proving that no other *Part A* agent is ever assigned a role.

We prove this specifically for the cell marked W, but it generalizes to the 4 other cells. Let A_W be the last *Part A* agent to reach the cell marked W, and let T be the round in which A_W enters the phase. Then A_W reaches the cell marked W in round $T + 14$.

The first *Part B* agent to reach the cell marked W is the first *Part B* Explorer (call it X). By Lemma 3.13, this is the fifth agent to begin executing *Part B*. No agent can begin executing *Part B* until at least round $T + 2$, and by Input Assumption 2, agents only enter the phase in even-numbered rounds. Therefore, X cannot enter the phase until at least round $T + 10$. By round $T + 14$ it is only partway through the path in Figure 3-3, so we know that A_W is never assigned the role of West Guide.

□

With this lemma, we can now prove that at least two agents of each role complete the *Allocation Phase*.

Theorem 3.15 (Guarantee 3)**.** *The Allocation Phase produces at least two agents of each role.*

Proof. Recall that by Input Assumption 4, $k' \geq 19$. By Lemma 3.14, we know that *Part A* produces exactly one agent of each role. Also by Lemma 3.14, we know that at most 14 agents execute *Part A*, so that means at least 5 agents execute *Part B*. Then by Lemma 3.13, *Part B* produces at least one agent of each role. □

For the algorithm to be guaranteed to locate the treasure, there must be at least two Explorers. However, for the algorithm to locate the treasure efficiently, we need the number of Explorers to be a constant fraction of k'. Thus, we have the following theorem.

Theorem 3.16 (Guarantee 4)**.** *Let X be the final Explorer to complete the Allocation Phase, and let T be the round in which it does so. Then after round T, there are at least $\lfloor (k'-9)/5 \rfloor$ and at most $\lfloor (k'-1)/5 \rfloor$ Explorers.*

Proof. By Lemma 3.14, at least 6 and at most 14 agents execute *Part A* of the *Allocation Phase*. The remaining agents all execute *Part B* of the *Allocation Phase*. From Lemma 3.13, we know that every fifth agent who executes *Part B* becomes an Explorer. So, the number of *Part B* Explorers after round T is at least $\lfloor (k-14)/5 \rfloor$ and at most $\lfloor (k-6)/5 \rfloor$.

Of the agents who execute *Part A*, exactly one becomes an Explorer. From this we have that there are at least $\lfloor (k'-14)/5 \rfloor + 1 = \lfloor (k'-9)/5 \rfloor$ and at most $\lfloor (k'-6)/5 \rfloor + 1 = \lfloor (k'-1)/5 \rfloor$ Explorers, as desired. □

As we will see in Section 3.3, it is essential that each agent knows whether it is the first agent of its role to enter the *Search Phase*. The reason that *Part A* is included is to ensure that the first agent of each role to complete the *Allocation Phase* is aware of that fact.

It is not immediately obvious how the agents executing the *Part A* and *Part B* protocols interact. In Lemmas 3.15 and 3.16, we show that the first *Part B* agent of each role is the first of its role to complete the *Allocation Phase*. We also show that each *Part A* agent completes the phase exactly two rounds after the first *Part B* agent of the same role. In Theorem 3.19, we show that the first *Part B* agent of each role is aware of the fact that it is the first by the round in which it completes the phase.

Lemma 3.17. *Let N_A be the Part A agent that becomes a North Guide, and N_B the first Part B agent to become a North Guide. Let T be the round in which N_B completes the Allocation Phase. Then N_A completes the Allocation Phase in round $T+2$. Symmetric statements hold true for South Guides and East Guides.*

Proof. We prove the statement for North Guides. Since East and South Guides follow symmetric protocols, the proof holds for them as well.

By Lemma 3.14, we know that N_A remains in the cell marked N in Figure 3-2 until it shares that cell with a *Part B* agent. Clearly N_B is the first *Part B* agent to reach the cell, so N_A is in its cell until N_B arrives there.

To show that N_A completes the *Allocation Phase* exactly two rounds after N_B, we show both agents executing their North Guide protocols in Figure 3-5. In Figure 3-5(a), we see N_A and N_B waiting in their positions. We see some other *Part B* agent, B, sharing N_B's cell. Assume that B is the fourth agent to share that cell with N_B, meaning that after that round N_B moves on in its North Guide protocol. Each subfigure represents the positions of the agents at the end of consecutive rounds. The agents are displayed in red in the round in which they complete the *Allocation Phase* and begin the *Search Phase* (and are not displayed in further rounds).

We can see that N_B completes the phase in Figure 3-5(e), and N_A completes the phase in Figure 3-5(g). Since these protocols are deterministic once N_B shares its cell for the fourth time, from this we can see that N_A always completes the *Allocation Phase* exactly two rounds after N_B, as desired.

□

Lemma 3.18. *Let X_A be the Part A agent that becomes an Explorer, W_A the Part A agent that becomes a West Guide, X_B the first Part B agent to become an Explorer, and W_B the first Part B agent to become a West Guide. Let T be the round in which X_B completes the Allocation Phase. Then we have*

1. *W_B completes the Allocation Phase in round $T + 1$.*
2. *X_A completes the Allocation Phase in round $T + 2$.*
3. *W_A completes the Allocation Phase in round $T + 3$.*

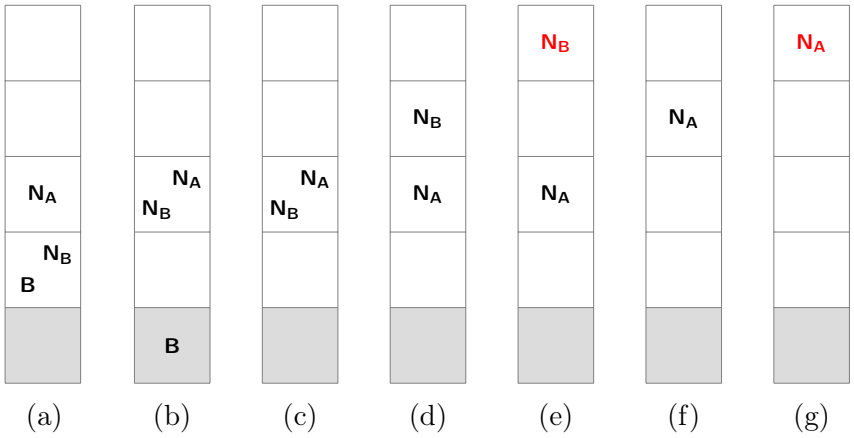


Figure 3-5: N_A is the *Part A* North Guide, and N_B is the first *Part B* North Guide. The center is shown in gray. In (a), we see that N_B shares a cell with some other *Part B* agent B, which is the fourth agent to share that cell with N_B. Each consecutive subfigure shows the agents at the end of consecutive rounds. In the round where an agent completes the phase it is shown in red and not shown again after that.

Proof. By Lemma 3.14, we know that X_A and W_A remain in the cells marked X and W in Figure 3-2, respectively, until they share those cells with *Part B* agents. Only *Part B* Explorers and West Guides enter those cells, so we know that X_A and W_A do not move until sharing their cells with X_B and/or W_B. The first agent to reach the cell marked W in Figure 3-3 becomes a West Guide, so that is W_B. The second agent to reach that cell moves along in the path and become an Explorer, and therefore must be X_B.

As in the proof of Lemma 3.17, we can prove this claim by demonstrating the protocol, which we do in Figure 3-6. Figure 3-6(a) shows the round before X_B reaches the cell where W_B waits, and Figure 3-6(b) shows the first round in which W_B and X_B share a cell.

Each subfigure shows the positions of the agents at the end of consecutive rounds. In the round where an agent completes the *Allocation Phase* and enters the *Search Phase*, the agent is shown in red (and then not showed in subsequent subfigures). We can see X_B completing the phase in Figure 3-6(e), W_B completing the phase in the following round (Figure 3-6(f)), X_A completing the phase in the following round

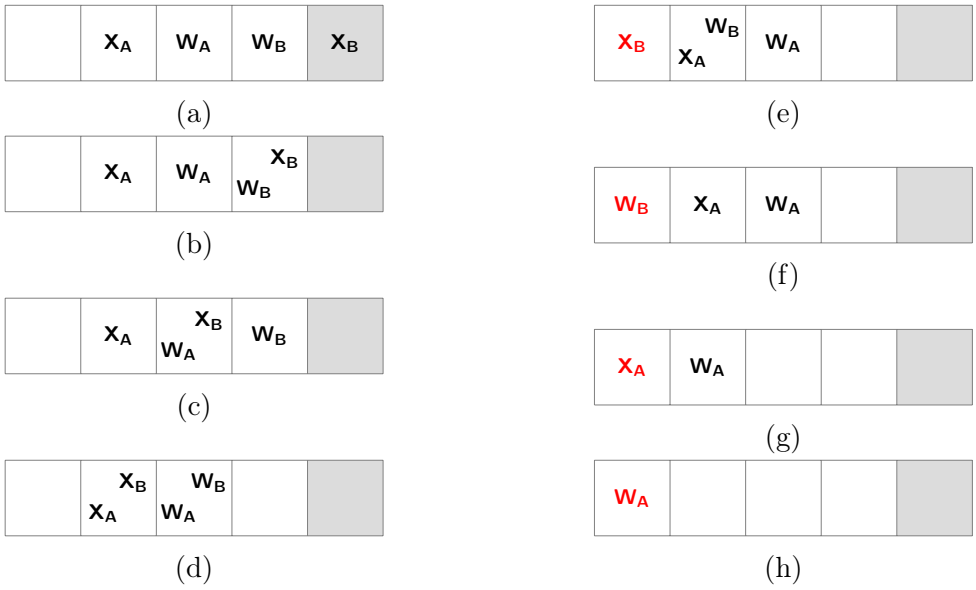


Figure 3-6: W_A and X_A are the *Part A* West Guide and Explorer, respectively, and W_B and X_B are the first *Part B* West Guide and Explorer. The center is shown in gray. In (a), we see X_B in the center, in the 7^{th} step along the path in Figure 3-3, while the other agents have been assigned their roles and are waiting in their cells. Each consecutive subfigure shows the agents at the end of consecutive rounds. In the round where an agent completes the phase it is shown in red and not shown again after that.

(Figure 3-6(g)), and W_A completing the phase in the round after that (Figure 3-6(h)).

□

Now that we have established that the first *Part B* agent of each role is the first agent of that role to complete the *Allocation Phase*, we need to ensure that those agents know that they are the first of their role to complete the phase, and that no later agents incorrectly believe that they are the first. We say that an agent knows that it is the first of its role to complete the phase if the $first$ variable in its state is TRUE. Note that all agents initially have $first$ = FALSE, and no $first$ variable is ever reset to FALSE once it has been changed. The following theorem establishes that the $first$ variables are correctly set.

Theorem 3.19 (Guarantee 5)**.** *Consider some agent A, and let T be the round in which A completes the Allocation Phase. In round T, the first variable in A's state*

is TRUE *if and only if no other agent of A's role completes the Allocation Phase before round T.*

Proof. First, we show that if A is the first agent of its role to complete the *Allocation Phase*, then it has the $first$ variable of its state set to TRUE when it completes the phase. Then we show that if some other agent of that role completes the phase first, then A never sets $first$ to TRUE. We specifically prove this for North Guides, although the proof generalizes to all other roles.

Say that A is the first North Guide to complete the phase. Then by Lemma 3.17, A is the first agent to become a *Part B* North Guide. By Lemma 3.17, we know that after A moves north once from the cell marked N in Figure 3-3, it is sharing its cell with the *Part A* North Guide, so it knows that it is the first North Guide to complete the phase.

Part A North Guides never modify the $first$ variable, so we know this theorem is satisfied if A is the *Part A* North Guide. Now we consider the case where A is a *Part B* North Guide other than the first one. By Lemma 3.17, the *Part A* North Guide leaves the cell marked N in Figure 3-2 two rounds after the first *Part B* North Guide leaves the cell. Clearly no other *Part B* North Guide can arrive in that cell by then, so we know that no other *Part B* North Guide shares that cell with the *Part A* North Guide.

By Lemma 3.14 no other *Part A* agent becomes a North Guide, so no *Part A* agent is in the cell marked N in Figure 3-2 once some *Part B* agent arrives there. Thus, we know that A is be alone when it reaches that cell, meaning it does not modify its $first$ variable. □

In the Lemma 3.18, we reason about the spacing between any two *Part B* agents of the same role. This will help to us to establish Guarantee 6 in Theorem 3.19, which says that there are at least 8 rounds between when any two agents (other than the first two) of a role complete the phase.

Lemma 3.20. *Consider some Part B agent B, and let T be the round in which B completes the Allocation Phase. Then no other Part B agent of the same role as B completes the phase before round $T + 10$.*

Proof. We prove the lemma for the case where B becomes a North Guide, but the proof for any other role is very similar. Consider the round T' which is the fourth round in which B shares the cell marked N in Figure 3-3 with some other agent. Any *Part B* North Guide completes the phase exactly 4 rounds after its last round in the cell marked N. We show that another agent cannot have shared that cell in 4 rounds until at least round $T' + 10$, which proves the lemma.

By Input Assumption 2, agents only enter the phase at most every other round. So the earliest another agent N can become a North Guide is round $T' + 2$. By the same reasoning N cannot share its cell for the first time until round $T' + 4$, for the second time until round $T' + 6$, for the third time until round $T' + 8$, and for the fourth time until round $T' + 10$, proving the lemma. □

Now we have the tools we need to establish the following theorem.

Theorem 3.21 (Guarantee 6)**.** *Let A be the first agent of its role to complete the Allocation Phase, and let T be the round in which it does so. Then*

1. *The second agent of that role completes the phase in round $T + 2$.*
2. *There are at least 8 rounds between when any other pair of agents of the role complete the phase.*

Proof. Part (1) follows from Lemmas 3.17 and 3.18. To prove part (2), we first consider the case where A is the *Part A* agent. By Lemmas 3.17 and 3.18, A completes the phase two rounds after the first *Part B* agent of its role. By Lemma 3.20, there are at least 10 rounds between when any two *Part B* agents complete the phase, so the next *Part B* agent does not complete the phase until at least 8 rounds after A. If

A is a *Part B* agent (other than the first one), then A completes the phase after the *Part A* agent of the same role, so Lemma 3.20 directly proves the lemma. □

As we will see in Section 3.3, it is important that each Explorer that enters the *Search Phase* is directly followed by a West Guide entering the phase in the following round. The following theorem establishes this fact.

Theorem 3.22 (Guarantee 7)**.** *Let X be an Explorer, and let T be the round in which X completes the Allocation Phase. Then a West Guide completes the phase in round $T + 1$.*

Proof. If X executes *Part A*, then the lemma is true by Lemma 3.18.

Consider the case where X executes *Part B*. For X to have become an Explorer, it must have reached the cell marked W in Figure 3-3 and found it occupied. Only a *Part B* West Guide could be waiting in that cell, so we know that X shares the cell with some West Guide W. Let T' be the round in which X and W share the cell.

Once it is in the cell marked W, X moves once more along the path in Figure 3-3, and then executes the *Part B* Explorer protocol, which takes two rounds, so X completes the phase in round $T' + 3$.

W had been waiting in the cell marked W to share in some round. Once it shares its cell with X, it pauses for one round, then moves west in the following three rounds and completes the phase in round $T' + 4$, which is the round after X. This proves the claim. □

In the Section 3.1 we proved that agents only enter the *Allocation Phase* on even-numbered rounds. Then we used the fact that no two agents enter the phase in consecutive rounds for proving some of our lemmas in this section. In the *Search Phase*, we need to directly use the fact that Explorers only enter the phase on even-numbered rounds. This is established by the following theorem.

Theorem 3.23 (Guarantee 8)**.** *Consider some agent X that becomes an Explorer in the Allocation Phase, and let T be the round in which X completes the Allocation Phase. Then T is even.*

Proof. By Input Assumption 2, X enters the *Allocation Phase* in an even-numbered round, so we just need to show that X spends an even number of rounds in the phase.

First consider the case where X executes *Part B*. Since X became an Explorer we know it followed the path outlined in Figure 3-3 to the end, which took an even number of rounds. After that it executed the *Part B* Explorer protocol, which requires only two rounds. Thus, X completes the phase in an even number of rounds.

If X executes *Part A*, then by Lemma 3.18 it completes the phase two rounds after some *Part B* Explorer, so we know that it also completes the phase in an even number of rounds. □

Our goal is to ensure that levels 0 through $D + 2$ are completely searched, as this guarantees that the treasure is located. As its name implies, most levels are searched during the *Search Phase*. However, agents in the *Search Phase* only search levels 4 and outward. The following theorem establishes that levels 0 through 3 are searched by agents in the *Allocation Phase*.

Theorem 3.24 (Guarantees 9 and 10)**.** *Let X be the first Explorer to complete the Allocation Phase, and let T be the round in which it completes the phase. Then after round T,*

1. *A North Guide, East Guide, and South Guide have completed the phase.*

2. *Every cell in levels 0 through 3 have been searched by some agent.*

Proof. By Lemma 3.17, X must have executed *Part B*. For some agent to have executed *Part B*, some other agent must have executed *Part A* and returned to the origin. Since X become an Explorer, we know that it reached the end of the path

in Figure 3-3. Thus, by round T, all the cells in the paths in Figures 3-2 and 3-3 have been searched. The only cells in levels 0 through 3 that are not covered by the union of those two paths are the cells exactly three moves north, east, and south of the center.

To have become a pending Explorer, X must have passed by the cell marked S in Figure 3-3 and found it occupied (say, by agent S), and passed by the cell marked W and found it occupied (say, by agent W). To be in its location, W must have passed by S, so then X is the second agent to pass S.

Consider the round in which S and X share a cell. In the next round, S moves south, then pause for a round, and then move south twice more, completing the phase. In this round X has just reached the cell marked X in Figure 3-2, so S completes the phase before X. Since S visits the cell 3 moves south of the center before completing the phase, we know that the cell three moves south of the center is searched before X completes the phase. A similar analysis shows that a North and an East Guide complete the phase before X, meaning that the cells three moves north and east of the center are also searched before round T. □

Finally, as we did in the *Separation Phase*, we outline the region that an agent in the *Allocation Phase* could be located. This will be important when we combine the phases to form RECTANGLE-SEARCH. The region is shown in Figure 3-7.

Lemma 3.25. *Let A be some agent in the Allocation Phase. Then A is at most three moves away from the center.*

Proof. An agent executing the *Allocation Phase* can be doing one of four things. Below, we argue that an agent doing any one of the four things is always at most three moves from the center.

1. The agent could be executing *Part A* and not be assigned any role.

In this case, the agent would be following the path outlined in Figure 3-2. We can clearly see that all cells along this path are within three moves from the center.

2. The agent could be executing *Part A* and be assigned some role.

   When a *Part A* agent becomes a Guide, it is two moves away from the center. The Guide protocol involves moving once, then moving again and in that round entering the *Search Phase*. Thus, while still in the *Allocation Phase*, it is at most three moves from the center.

   When a *Part A* agent becomes an Explorer, it is three moves away from the center, but in the round when an agent moves in the *Part A* Explorer protocol, it enters the *Search Phase*, so it is always within three steps from the center while in the *Allocation Phase*.

3. The agent could be executing *Part B* and not be assigned any role.

   The agent would be following the path outlined in Figure 3-3. As in (1), we can see that all cells along this path are within three moves from the center.

4. The agent could be executing *Part B* and be assigned some role.

   Similar to (2). When a *Part B* agent becomes a Guide, it is one move from the center. The Guide protocol requires three moves, but in the round of the third move the agent enters the *Search Phase*, so while in the *Allocation Phase* it is always at most three moves from the center. When a *Part B* agent becomes an Explorer, it is two moves from the center. After that it moves twice, but in the round of the second move it enters the *Search Phase*.

□

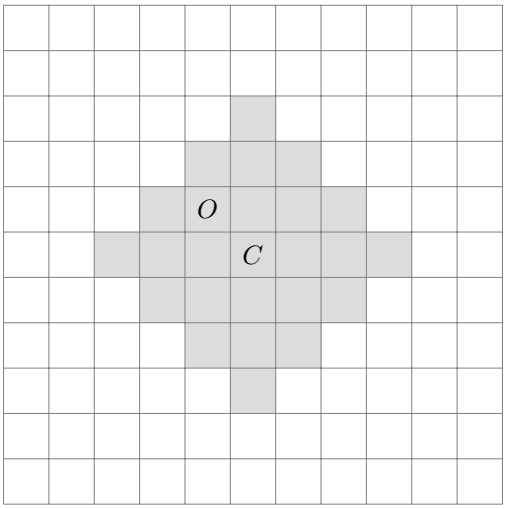


Figure 3-7: The gray region indicates the cells in which an agent from the *Allocation Phase* could be located. The cell marked O is the origin, and the cell marked C is the center.

3.2.3 Analysis

All agents other than the *Part B* Explorers have a part of their protocol where they have to wait for other agents. Thus, only the *Part B* Explorers spend a predetermined number of rounds in the *Allocation Phase*. Since we perform our eventual analysis of RECTANGLE-SEARCH by analyzing the behavior of Explorers, it turns out we only need to bound the number of rounds Explorers spend in the phase.

Lemma 3.26. *Consider an agent A who became an Explorer in Part B of the Allocation Phase. Then A spends* 10 *rounds in the Allocation Phase.*

Proof. To have become a *Part B* explorer, A must have followed the path in Figure 3-3 and reached the end of the path. This took exactly 8 rounds. After that, the agent became an Explorer. The *Part B* Explorer protocol involves only two rounds (two moves west), so the total number of rounds A spends in the *Allocation Phase* is 10. □

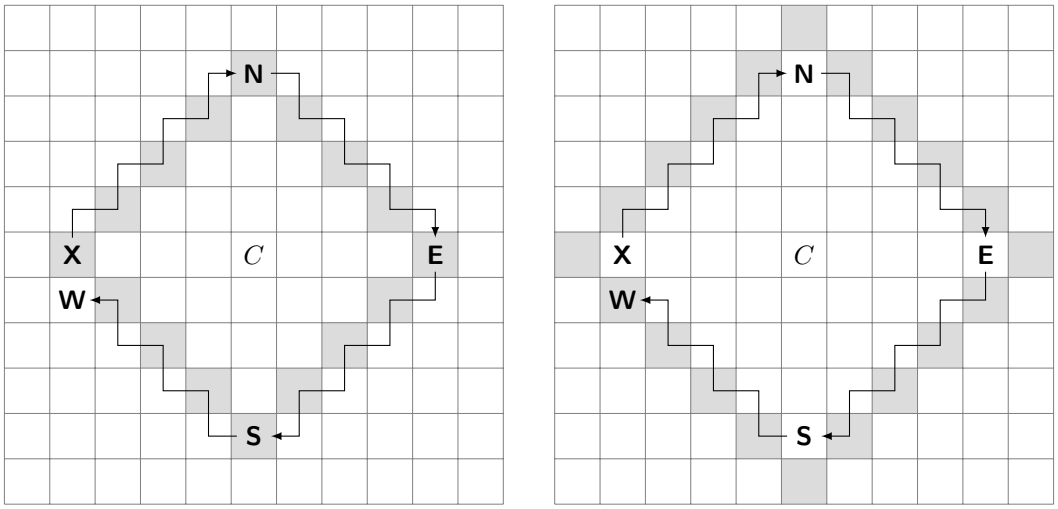
3.3 Search Phase

In the *Search Phase*, the agents work collaboratively to search the grid from the center outwards. Recall that we defined *level* d to be the set of all cells exactly d moves from the center. The Explorers progressively explore the levels. To explore a level, an Explorer starts on the west axis, then alternates between moving north and east until it reaches the north axis. It knows when it has reached the north axis because a North Guide will be positioned there. It then repeats this process in the next three quadrants, each time finding a Guide in the appropriate place so that it knows when to switch directions. As we will see, the West Guide will actually be waiting directly below the west axis for the Explorer to return.

The idea of exploring levels and using Guides to mark the axes for Explorers was first presented in [2]. Since our models allows less communication than [2], some details had to be modified. For example, in the model from [2], a Guide can sense whether it is sharing its cell with another Guide or an Explorer, but our model does not allow agents to sense this.

The path of an Explorer exploring level 4 is shown in Figure 3-8. As we can see from the figure, an Explorer exploring level 4 also explores almost all of level 5. In fact, the only level 5 cells that it does not explore are those located on the axes. As we will see, the cells on the axes are explored by the Guides, so it is not necessary for an Explorer to ever explore level 5. In general, if an Explorer explores level d, then it is not necessary for another Explorer to explore level $d + 1$. In the *Search Phase*, the first Explorer explores level 4, the next Explorer level 6, and so on. When an Explorer finishes exploring a level, it moves outwards and explores a new level.

In this section, we describe exactly how the agents coordinate these efforts in order to ensure that all the levels are explored. As in Sections 3.1 and 3.2, we begin by describing the algorithm to be performed. Then we argue why this algorithm results in the desired behavior, and reason about the number of rounds it takes the agents



(a) Level 4 cells highlighted (b) Level 5 cells highlighted

Figure 3-8: Both subfigures display the path an Explorer X would take to Explore level 4. N, E, S, and W are North, East, South, and West Guides, shown in the positions that they must be in for X to properly explore the level. The cell marked C is the center. In (a) the level 4 cells are highlighted, and in (b) the level 5 cells are highlighted, demonstrating that when level d is explored, almost all the level $d+1$ cells are also explored.

to explore levels 4 through d completely.

Input Assumptions

1. Every agent that enters the phase is assigned one of five roles: North Guide, East Guide, South Guide, West Guide, or Explorer.

2. All North Guides enter the phase four cells north of the center, East Guides four cells east of the center, South Guides four cells south of the center, and West Guides and Explorers four cells west of the center.

3. At least two agents of each role enter the phase.

4. Every agent knows whether it is the first of its role to enter the phase.

5. The second agent of any given role enters the phase exactly two rounds after the first agent of the role, and there are at least 8 rounds between when any

two other agents of that role enter the phase.

6. A West Guide always enters the phase in the round after an Explorer completes the phase.

7. Every Explorer enters the phase in an even-numbered round.

8. Before the first Explorer enters the phase, a North Guide, an East Guide, and South Guide have entered the phase.

Guarantee

1. For any even $d \geq 4$, there is some round in which an Explorer completes exploring level d, and at that time levels 4 through d have been searched completely.

3.3.1 The Algorithm

Informal Description

In this phase, each agent's state has a variable *status*, which is either *exploring* or *between*. An agent begins the phase with a status of *between*. Afterwards, it alternates between having a status of *exploring* and a status of *between*.

For simplicity, we refer to any cell that is an even number of moves from the center as an *even cell* (and similarly for an *odd cell*). Now, if an agent has just entered the phase and is the first of its role to do so, it immediately changes its status to *exploring*. In any other situation, an agent with a status of *between* moves in its designated direction until it is in an even cell with another agent. It then continues moving in its direction. If it ever reaches an odd cell that is occupied, it restarts its search for an occupied even cell. After finding an occupied even cell, it continues moving in its direction. When it finds an even cell that is unoccupied, it switches to a status of *exploring*.

The behavior of an agent with a status of *exploring* differs between the different roles. North, East, and South Guides behave the same, but West Guides and Explorers each have different protocols.

North, East, and South Guides: With a status of *exploring*, the agent remains in its cell until it shares it in 3 consecutive rounds. Then, as soon as it senses that it is alone in its cell, it switches to a *between* status.

West Guide: When the West Guide has a status of *exploring*, it remains in its cell until it has shared its cell for 2 consecutive rounds. Then it moves south once. It remains there until there is a round where it shares its cell. After that round, it moves north once and then does not move for a round. If it is alone in the round that it does not move, it immediately switches to a status of *between*. Otherwise, it waits in its cell until a round where it detects that it is alone. Then it does not move for one round, and then switches to a status of *between*.

Explorer: Say that an Explorer switches to a status of *exploring* while in a cell on level d. Then we say that the agent is *exploring level* d (until it switches back to having a status of *between*). With a status of *exploring*, the Explorer begins by moving north, then east, then north again (the reason for these three initial steps will become clear later). After that, it alternates between moving east and north until it shares its cell with another agent for a round.

After detecting it is not alone, it does not move for two rounds. Then, it alternates between moving east and moving south. After reaching another agent, it does not move for two rounds again and then alternates between moving south and west. Once it reaches another agent, it again does not move for two rounds and then alternates between moving west and north until it reaches another agent. After that, it moves

west once and north once more. Once it is alone in its cell, it switches to a status of *between*.

Formal Description

Note that in order to define the state transition function δ-SEARCH for this section, we use helper functions δ-NES-SEARCH, δ-W-SEARCH, and δ-X-SEARCH.

Each state $s \in S$ contains the following variables. The variables that are not used in this phase are omitted here.

phase: initially SEARCH (does not change)
role: member of {NORTH, EAST, SOUTH, WEST, EXPLORER},
value set in previous phase
$first$: boolean, value set in previous phase
status: member of {BETWEEN, EXPLORING}, initially BETWEEN
$found$: boolean, initially FALSE
wait: boolean, initially FALSE
down: boolean, initially FALSE
even: boolean, initially TRUE
quad: integer in $\{1, \ldots, 4\}$, initially 1
rounds: integer in $\{0, \ldots, 4\}$, initially 0
shared: integer in $\{0, \ldots, 3\}$, initially 0
count: integer in $\{0, \ldots, 3\}$, initially 0

```
δ-SEARCH(s, alone, origin)
    if s.first
        s.first ≔ FALSE
        s.even ≔ FALSE
        s.status ≔ EXPLORING
    if s.status = BETWEEN
        if not alone
            if s.even
                s.found ≔ TRUE
            else
                s.found ≔ FALSE
        elseif s.found and s.even and alone
            s.found ≔ FALSE
            s.wait ≔ FALSE
            s.even ≔ TRUE
            s.shared ≔ 0
            s.count ≔ 0
            s.status ≔ EXPLORING
        s.even ≔ not s.even
    if s.status = EXPLORING
        if s.role = NORTH or EAST or SOUTH
            s ≔ δ-NES-SEARCH(s, alone, origin)
        elseif s.role = WEST
            s ≔ δ-W-SEARCH(s, alone, origin)
        elseif s.role = EXPLORER
            s ≔ δ-X-SEARCH(s, alone, origin)
    return s
```

```
δ-NES-SEARCH(s, alone, origin)
    if not alone
        if s.shared < 3
            s.shared ≔ s.shared + 1
    else
        s.shared ≔ 0
    if s.shared = 3
        s.wait ≔ TRUE
    if s.wait and alone
        s.status ≔ BETWEEN
    return s
```

δ-W-SEARCH$(s, alone, origin)$

```
s.down := FALSE
if not alone and s.shared < 3
    s.shared := s.shared + 1
    if s.shared = 2
        s.down := TRUE
if s.count ≥ 2 and not s.wait
    if alone
        s.status := BETWEEN
    else
        s.wait := TRUE
if alone
    s.wait := FALSE
if s.shared = 3 or (s.count > 0 and < 3)
    s.count := s.count + 1
return s
```

δ-X-SEARCH$(s, alone, origin)$

```
s.even := not s.even
if s.rounds < 4
    s.rounds := s.rounds + 1
if s.rounds = 4 and not alone and s.shared < 3
    s.shared := s.shared + 1
if s.shared = 3
    s.shared := 0
    s.quad := s.quad + 1
if (s.quad = 4 and s.shared = 1) or (s.count > 0 and < 4)
    s.shared = 0
    s.count := s.count + 1
if s.count ≥ 3 and alone
    s.even := TRUE
    s.quad := 1
    s.rounds := 0
    s.status := BETWEEN
return s
```

```
M-SEARCH(s)
    if s.status = BETWEEN
        if s.role = EXPLORER
            return WEST
        else
            return s.role
    elseif s.status = EXPLORING and s.role = WEST
        if s.down
            return SOUTH
        elseif s.count = 2
            return NORTH
    elseif s.status = EXPLORING and s.role = EXPLORER
        if s.shared = 0 and s.count < 3
            if s.count > 0
                if s.even return WEST
                else return NORTH
            elseif (s.quad = 1 and s.even) or (s.quad = 4 and not s.even)
                return NORTH
            elseif (s.quad = 1 and not s.even) or (s.quad = 2 and s.even)
                return EAST
            elseif (s.quad = 2 and not s.even) or (s.quad = 3 and s.even)
                return SOUTH
            elseif (s.quad = 3 and not s.even) or (s.quad = 4 and s.even)
                return WEST
    return NONE
```

3.3.2 Correctness

In this section we prove that the protocol outlined above ensures that *Search Phase* agents locate the treasure if it is not located by agents in previous phases. We begin by analyzing the number of rounds that it takes an agent in the *Search Phase* to search all level d cells when exploring level d, assuming that the appropriate Guides are in place so that the Explorer correctly switches directions when arriving at an axis. While Lemma 3.27 is also used in the analysis section, we will see that is necessary to prove the correctness of this phase.

Lemma 3.27. *Consider an Explorer X that switches to an exploring status in level d in round T. Then if there are North, East, and South Guides positioned in the level d cells on their respective axes when X reaches those cells, then X is in the cell south*

of the level d west axis cell in round $T + 8d + 5$.

Proof. First, we note that only a level d Explorer could be located in a level d or $d+1$ cell other than those cells on each of the axes or directly south of the west axis.

X begins by alternating between moving north and east until it shares its cell for a round. By the above statement it does not share its cell until it reaches the north axis. This takes $2d$ rounds. We are assuming that a North Guide is located in the level d cell on the north axis, so X waits on the north axis for 2 rounds.

Then it repeats the process, spending $2d$ rounds to reach the east axis, 2 rounds waiting on the east axis, $2d$ rounds to reach the south axis, and 2 rounds waiting on the south axis. From the level d cell on the south axis, it takes $2d-1$ rounds to reach the cell south of the level d cell on the west axis, for a total of $8d + 5$ rounds.

□

The next theorem ensures that some Explorer searches each level, and that the levels are explored from level 4 outwards. This ensures us that by the round in which some Explorer completes exploring level $D + 2$, the treasure must have been located.

Theorem 3.28. *Consider some Explorer X that completes exploring level d in round T for some even $d \geq 4$. Then in round T, levels 4 through d have been completely searched by Search Phase agents.*

Proof. For brevity, we define X_d to be the first Explorer to begin exploring level d. Similarly, define N_d, E_d, S_d, and W_d as the first North, East, South, and West Guides, respectively. We define the cell n_d to be the level d cell on the north axis, and similarly for the cells e_d, s_d, and w_d.

To prove this lemma, we inductively prove the following 8 part claim. We will see that it is necessary to prove all 8 parts together due to dependence between the parts. For any even $d \geq 4$,

1. N_d does not switch from an *exploring* status in cell n_d before sharing its cell with X_d.

2. If N_d switches from an *exploring* to *between* status in cell n_d in round T, then in round T, n_d is not part of a chain of three consecutive cells on the north axis containing North Guides with statuses of *between*.

3. N_d and N_{d+2} switch to statuses of *exploring* in cells n_d and n_{d+2} before X_d reaches the north axis.

4. X_d is the first Explorer to reach cell w_d.

5. No Explorer other than X_d or West Guide other than W_d ever explores level d.

6. W_d is in the cell south of w_d in the round when X_d reaches that cell, and no other agents are there.

7. If an Explorer X is located in cell w_d at time T, then T is even.

8. If an Explorer X is located in cell w_d at time T, then a single West Guide sharing X's status is located in cell w_d at time $T+1$. Furthermore, W is alone at time $T+1$ if and only if X is alone at time T.

Combining parts (1), (3), and (6) show that when an Explorer explores a level, all the Guides will be in place, so Explorers always finish exploring levels that they start exploring. Part (2) helps us to prove part (1). Part (4) shows that no level can be skipped. Part (5) shows that the agents search the space efficiently. This is useful in the analysis section, but is included here because it is necessary for the proof of part (6). It is not immediately obvious why parts (7) and (8) are useful, but we will see that they are needed for the proofs of parts (5) and (6).

We begin by showing that all the statements hold true for $d = 4$. For this, we can prove each of the parts separately.

1. N_4 cannot switch from the *exploring* status until sharing the cell n_4 for 3 consecutive rounds. The only agents who can ever enter the cell n_4 are North Guides and level 4 Explorers. All North Guides other than the first North Guide to enter the *Search Phase* enter the phase in the cell n_4 and then move north in the following round. By Input Assumption 5, no two North Guides enter the phase in consecutive rounds, so other North Guides cannot cause N_4 to share its cell for 3 consecutive rounds, and therefore it remains there until X_4 arrives.

2. For N_4 to be part of a chain of 3 consecutive North Guides in *between* states, other North Guides would have to enter the phase in either rounds $T-1$ and $T-2$, $T-1$ and $T+1$, or $T+1$ and $T+2$. By Input Assumption 5, for any two North Guides not including N_4 there are at least 8 rounds between when they enter the phase, so clearly this cannot be the case.

3. The first agent of each role to enter the *Search Phase* immediately begins exploring level 4. So X_4 and N_4 are the first Explorer and North Guide to enter the phase, respectively. Let T_X be the round in which X_4 enters the phase and T_N the round in which N_4 enters the phase. By Input Assumption 8, $T_N < T_X$.

   X_4 reaches the north axis in round $T_X + 8$. Since $T_N < T_X + 8$, N_4 switches to the *exploring* status before X_4 reaches the north axis. By Input Assumption 5, another North Guide enters the phase in round $T_N + 2$. By part (1), N_4 will still be located in the cell n_4 at that time. So, the North Guide will share the cell n_4 with N_4 and then move north in search of an unoccupied even cell. It will find this in the cell n_6, so it will switch to a *exploring* status there in round $T_N + 4$.

4. Clearly the first Explorer to enter the phase will be the first Explorer to reach the cell w_4. The protocol dictates that the first Explorer to enter the phase explores level 4.

5. Any Explorer or West Guide that is not the first to enter the phase must find an occupied even cell and then an unoccupied even cell further from the origin before it can explore a level, so it clearly cannot explore level 4.

6. Let T be the round in which X_4 enters the phase. Then by Input Assumption 6 W_4 enters the phase in round $T+1$. Its protocol dictates that it remains in the cell w_4 until sharing it in 2 consecutive rounds. By Input Assumptions 6 and 5, an Explorer and West Guide enter the phase in rounds $T+2$ and $T+3$ respectively, so they share W_4's cell in those rounds. So then W_4 moves south in round $T+4$. Only a level 4 Explorer or West Guide could ever reach the cell directly south of w_4. By part (5) there are no other level 4 West Guides, so W_4 remains south of w_4 and alone until X_4 arrives.

7. By Input Assumption 7, X enters the phase in an even-numbered round. We claim that X is only located in the cell w_4 in the round in which it enters the phase. If X is the first Explorer to enter the phase it switches to an *exploring* status and moves north the round after it enters the phase. By part (6), X reaches a West Guide in the cell south of w_4, after which it moves west and then north, so that when it returns to the west axis it is in cell w_{d+1}. If X is not the first Explorer to enter the phase it remains with a *between* status moves west in the next round.

8. If X is the first Explorer to enter the phase, then by Input Assumption 6 the first West Guide enters the phase in the following round, and both have status *exploring*. It is easy to see that both agents are alone in the rounds they enter the phase, so this part is satisfied for this case.

   If X is not the first Explorer to enter the phase, then it is slightly more complicated. By Input Assumption 6, some West Guide W always enters the phase in the round after X, and since neither are the first of their role to enter the

phase, they both have status *between*. It remains to show that X is alone at time T if and only if W is alone at time $T+1$.

The only agent other than X and W who could be in the cell w_4 is W_4. We show that if W_4 is located in the cell w_4 at time T, then it is also located there at time $T+1$. First we show that W_4 cannot exit the cell in round $T+1$. If W_4 has not yet moved south when X shares its cell, then X must be the first agent to share its cell, so W_4 remains in the cell for at least one more round. If W_4 had already moved south and then north again, then it is waiting to be alone to switch to a *between* status, so does not move in round $T+1$.

The only round in which W_4 enters the cell is when it moves north back to the west axis after having shared the cell south of w_4 with X_4. But X_4 begins exploring on an even round, and by Lemma 3.27 reaches W_4 below the west axis in an odd-numbered round. Since W_4 moves north in the next round, it enters w_4 in an even-numbered round. Since $T+1$ is odd, it cannot enter in round $T+1$, completing the proof of this part.

Next we need to show the inductive step. For this part, we cannot separate all of the parts as we did for the base case, because many of them are dependent on each other. We assume that the claim holds for $d-2$, and prove that it holds for d.

1. For N_d to switch from a status of *exploring* in cell n_d, it must share the cell n_d in 3 consecutive rounds. Assume for contradiction that N_d switches from a status of *exploring* before sharing cell n_d with X_d. We know that only other North Guides could ever be located in that cell, and that no other North Guide remains in that cell for more than a single round while it is occupied by N_d. So then it must be the case that 3 distinct North Guides enter the phase in consecutive rounds.

   We know that no North Guides can enter the phase in consecutive rounds. So

then the 3 North Guides cannot all be in their first *between* status. But by assumption (2), any North Guide that switches from an *exploring* status to a *between* status doesn not form a chain of three consecutive North Guides, presenting a contradiction.

2. Let T_{d-2} be the round in which N_{d-2} switches from a *exploring* status to a *between* status in n_{d-2}. Then at the latest, X_{d-2} entered the north axis in round $T_{d-2} - 3$.

   We claim that X_d cannot reach the north axis until at least round $T_{d-2} + 5$. To see why, consider the round T' in which X_{d-2} began exploring level $d-2$. Combining assumptions (4) and (7), the next explorer furthest from the center could be located in at most cell w_{d-4}. But then X_{d-2} reaches the north axis in round $T' + 2(d-2)$, and so that next Explorer cannot possibly reach the north axis until round $T' + 4 + 2d$, or 8 rounds after X_{d-2} reaches it.

   At time $T_{d-2}+5$, if N_d is still has a status of *between*, then it is in cell n_{d+3}. By assumption (2), either cell n_{d+2} or n_{d+1} does not contain a North Guide with a *between* status in that round, so N_d does not join a chain of three consecutive *between* status North Guides including N_{d-2}.

   A similar analysis shows that N_d cannot form a chain of length 3 with $N_{d'}$ for any $d' < d-2$. By Input Assumption 6, we know that two North Guides with their first *between* status cannot form a chain of length 3 with N_d, completing the proof.

3. Let T be the round in which X_{d-2} switches to an *exploring* status in cell w_{d-2}. It takes $2(d-2)$ rounds for X_{d-2} to reach the north axis. By assumption (3), we know that by round $T+2(d-2)$, N_{d-2} is in an *exploring* state on level $d-2$. So, when X_{d-2} reaches the north axis it waits there for 2 rounds. Thus, X_{d-2} leaves the north axis in round $T + 2(d-2) + 2 = T + 2d - 2$. If N_{d-2} is alone

at time $T + 2d - 2$, it switches to a *between* status.

If it is not alone, it must be sharing its cell with some other North Guide in a *between* state, so without loss of generality we can assume that N_{d-2} was alone and entered a *between* state.

By assumption (3), N_d had a status of *exploring* in cell n_d by round $T+2(d-2)$, and by part (1) remains in the cell until X_d shares its cell. By the same reasoning as in part (2), X_d cannot begin exploring level d until at least round $T + 4$, so it cannot reach the north axis until round $T + 4 + 2d$.

So, when N_{d-2} reaches cell n_d in round $T + 2d$, it finds it occupied by N_d and continues to move north in search of an unoccupied even cell. It reaches cell n_{d+2} in round $T + 2d + 2$. If it is alone at that time, it switches to a status of *exploring*. Otherwise, there was already some North Guide located there in an *exploring* state. So, N_{d+2} is in an *exploring* state in the level $d + 2$ north axis cell by round $T + 2d + 2$.

4. We claim that X_{d-2} cannot return to the west axis before another Explorer reaches the cell w_{d-2}. By assumption (6), W_{d-2} moved south from cell w_{d-2} before X_{d-2} reached that cell. To have moved south, W_{d-2} must have shared its cell for two consecutive rounds. By assumption (8), it must have first shared its cell with an Explorer X, and then with a West Guide W.

   After sharing the cell w_{d-2} with W_{d-2}, X begins exploring the next unoccupied even cell that it reaches, as long as it does not reach an occupied odd cell before that. Since X is the first agent to reach the cell w_{d-1}, it is alone there. So then it reaches the cell w_d and also be alone, and enters an *exploring* state. So then $X_d = X$. A similar analysis shows that $W_d = W$, proving this part.

5. Consider an Explorer X who reaches level d after X_d starts exploring the level. We show that X does not explore level d. If W_d is in cell w_d when X arrives,

then clearly X does not explore level d. Otherwise, we claim that when X reaches cell w_d, it has $found = \text{FALSE}$, so it does not explore the level.

Since X_d and W_d moved from level 4 to level d on the west axis, any West Guide that was on the west axis would have already moved south by the round in which X arrives there. Say that X shares a cell with some West Guide $W_{d'}$ in level $d' < d$. Then W' must have just moved north and be awaiting becoming alone to switch to a *between* status. Since $W_{d'}$ only moves north after sharing its cell with $X_{d'}$, we know that $X_{d'}$ moves into the cell $w_{d'+1}$ in the round after $W_{d'}$ moves north and does not move until it is alone. So when X moves west once it shares the odd cell $w_{d'+1}$ with $X_{d'}$ and set $found = \text{FALSE}$.

6, 7, 8. Consider some Explorer X that does not explore level $d-2$. Say that X is in the cell w_{d-2} at time T. By assumption (7) T is even. Since X does not explore that level, it remains with a status of *between* and moves west in the next round. By assumption (8), there is be a West Guide W in the cell w_{d-2} at time $T+1$ that also does not explore level $d-2$. So then at time $T+2$, X is in the cell w_d, satisfying part (7) for when X does not explore level $d-2$. The same reasoning an in part (8) of the base case shows that X shares the cell w_d at time $T+2$ if and only if W shares the cell w_d at time $T+3$, satisfying part (8) for when X does not explore level $d-2$.

Now, we need to show that by the round in which X_d reaches the cell south of w_d, W_d (and no other agent) is there. By part (5), no Explorer ever begins exploring level d after X_d does, so W_d is the only West Guide that could possibly be in the cell south of w_d.

Let T be the round in which X_{d-2} begins exploration of level $d-2$. Then by Lemma 3.27 and assumption (6), X_{d-2} and W_{d-2} share the cell south of w_{d-2} at time $T + 8(d-2) + 5$ (an odd round). In the following round, W moves north

and X moves west. In the found after that, W does not move, and X moves north.

This places X_{d-2} in an odd cell in an odd round and W_{d-2} in an even cell in an even round. If X_{d-2} is sharing a cell in this round, then by our proof of part (8) above for Explorers that do not explore level $d-2$, it must be sharing its cell with some other Explorer, and in the following round shares its cell with some West Guide. Thus, X_{d-2} waits an even number of rounds before switching to a *between* status. Thus, it switches to a *between* status in an odd-numbered round, meaning it reaches level d in an even-numbered round, proving part (7).

In the round when X_{d-2} returns to the west axis, W_{d-2} is in the cell east of X_{d-2}, so it must become alone in the round before X_{d-2}. So, by pausing for a round before switching to a *between* status, it switches to a *between* status in the same round as X_{d-2}, and reach the cell w_d a round later, proving part (8).

If X_{d-2} is alone upon reaching the west axis, it would immediately switch to a *between* status, and would arrive in the cell w_d round $T + 8(d-2) + 5 + 3 = T + 8d - 8$. If X_{d-2} was not alone, it was sharing its cell with some other agent that would then arrive in the cell w_d in round $T + 8d - 8$. Similar reasoning shows that either W_{d-2} or some other agent arrives in the cell w_d in round $T + 8d - 7$.

If W_d was still in the cell w_d at time $T + 8d - 7$, then it certainly moves south in the next round. By Lemma 3.27, X_d takes $8d + 5$ rounds to reach the cell south of the level d west axis cell, so W_d is there before that, proving part (6).

□

Next, as Sections 3.1 and 3.2, we outline the region in which a *Search Phase* agent could be located. The region is displayed in Figure 3-9.

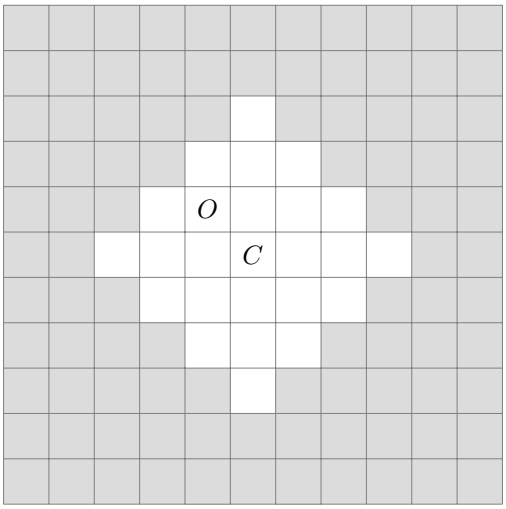


Figure 3-9: The gray region indicates the cells in which an agent from the *Search Phase* could be located. The cell marked O is the origin, and then cell marked C is the center. Note that the gray region includes all the cells outside grid displayed here.

Lemma 3.29. *Let A be some agent in the Search Phase. Then A is at least four moves from the center.*

Proof. By Input Assumptin 2, each agent enters the phase exactly four moves from the center. No Guide ever moves in the opposite of its cardinal direction, so it can never get any closer to the center, so it is always at least four moves from the center.

With a status of *between*, Explorers only move west. So, if an Explorer switches to a status of *between* in the region, then it remains in the region while having the status of *between*. With a status of *exploring*, when exploring *level* d, the Explorer never moves into any cell less than d moves from the origin. Since the innermost level explored is level 4, no agent with a status of *exploring* is ever less than four moves from the center. □

3.3.3 Analysis

Since agents are all in the phase for different numbers of rounds, it is difficult to upper bound the number of rounds that an agent spends in the *Search Phase*. So, we

focus on one particular Explorer, and upper bound the number of rounds that that Explorer could spend in the phase before levels 4 through d have been completely searched, for any arbitrary $d \geq 4$.

Lemma 3.30. *Let T be the round in which some Explorer completes exploring level d for some even $d \geq 4$. Say that there are i Explorers in the Search Phase after T rounds, and let X be the last Explorer to enter the phase before the end of round T. Then the number of rounds that X spends in the phase before levels 4 through d have been completely searched is $O(d + d^2/i)$.*

Proof. When X enters the *Search Phase*, it moves out to find the first unexplored level. Say that this is level l. Once there, it explores that level. By Lemma 3.27 it takes $O(l)$ rounds to explore level l. After exploring this level, it passes by all the levels that are currently being explored by another Explorer, and explores the next level. Since Explorers explore every other level, and there are $i - 1$ other Explorers, this next level is level $l + 2i$. After exploring that level, it explores level $l + 4i$, and so on. This process is repeated at most $d/2i$ times before level d is explored. So, the total number of rounds is given by

$$\begin{aligned} l + \sum_{z=0}^{d/(2i)} O(l + z \cdot (2i)) + 2i &= l + O\left(\sum_{z=0}^{d/i} l + zi\right) \\ &= l + O\left(\sum_{z=0}^{d/i} l + i\sum_{z=0}^{d/i} z\right) \\ &= l + O((d/i) \cdot l + d^2/i) \end{aligned}$$

To simplify this further, we claim that $l = O(d)$. To see why, we note that if $l > d$, then it means that by the round in which X starts exploring its first level, some other Explorer is already exploring level d. That Explorer completes exploring level d within $O(d)$ rounds, so X cannot possibly start exploring a level more than $O(d)$ moves from the center. With this in mind, the above sum simplifies to

$$O(d + d^2/i + d^2/i) = O(d + d^2/i),$$

as desired. □

3.4 Putting it All Together

In this section, we show how to combine the *Separation*, *Allocation*, and *Search Phase*s to form the algorithm Rectangle-Search. We prove that Rectangle-Search eventually locates the treasure with probability 1, and reason about the expected number of rounds until it does so.

In Sections 3.1, 3.2, and 3.3, we analyzed the behavior of the agents in each phase under the assumption that the agents' behavior would not be impacted by agents in the other phases. The next three lemmas, which we will collectively refer to as the *Isolation Lemmas*, prove that this assumption is valid.

Before we begin, we define notation that is used in all three Isolation Lemmas. Consider some fixed execution α of Rectangle-Search, and let A be an agent in the execution. For any $i \geq 0$, let c_i be the cell in which A is located at time i in α.

In each Isolation Lemma, we prove that when A is in any given phase, it would have occupied the same sequence of cells if the agents from the other two phases had not been present. Recall that we defined a *ghost* to be an agent which executes the algorithm normally, but the other agents do not detect its presence. We will define modified versions of α in which agents in a specific phase treat agents in the other phases as ghosts. We will show that the sequence of cells which an agent occupies in one of these modified executions is equivalent to the sequence of cells it occupies in α.

In Sections 3.1, 3.2, and 3.3, we effectively proved the correctness of each phase in this modified execution. By proving that the cells occupied by each agent in these

modified executions are identical to the cells occupied by the agents in α, we see that the guarantees of each phase are met when we combine all three phases.

Lemma 3.31 (Isolation Lemma 1)**.** *Let α' be a fixed execution that is identical to α except agents in the Separation Phase treat agents in other phases as ghosts. Let T be the round in which A completes the Separation Phase in α'. Let c'_i be the cell occupied by A at time i in α' for some i, $0 \leq i \leq T$. Then we have $c_i = c'_i$.*

Proof. We induct on i to prove our claim. Since all agents are at the origin at time 0, we know that if $i = 0$ we have $c_i = c'_i$. If $i > 0$, assume that we have $c_{i-1} = c'_{i-1}$. We prove that $c_i = c'_i$. To do so, we prove that the move chosen by A in round $i - 1$ of α is not impacted by the presence of agents in other phases, and therefore it must be the same as the move chosen by A in round $i - 1$ of α'.

We begin by arguing that in α, A never performs loneliness detection while sharing a cell with a member of the *Allocation Phase*, which clearly implies that A makes the same moves it would have made if those agents had not been present. By Lemmas 3.4 and 3.25, the only cells that both A and an agent in the *Allocation Phase* could ever occupy are $(-1, 0)$ and the origin. However, since A is only at the origin in the *Separation Phase* at time 0, we only need to consider the cell $(-1, 0)$.

There are two times at which A could occupy the cell $(-1, 0)$. It occupies the cell at time 1, but moves west deterministically in the following round. It also occupies the cell when moving east back towards the origin, but once it starts moving east it never uses its ability to detect its loneliness. So clearly no *Allocation Phase* agent affects the cells occupied by A.

Next we show that in α, agents from the *Search Phase* can never impact the moves made by A. In the *Search Phase*, North, East, and South Guides never leave their respective axes, and West Guides only ever occupy cells on the west axis and cells one move south of the west axis. By Lemma 3.4, an agent in the *Separation Phase* can never occupy any of those cells. Explorers with a status of *exploring* are always

located on the west axis. So we just need to show that an Explorer with a status of *between* cannot impact the moves made by A.

Say that A shares cell c_{i-1} with some Explorer X at time $i-1$. We claim that no *Search Phase* agent is located in cell c_{i-1} at times $i-2$ or i. Since Explorers with an *exploring* status move every round until returning to the west axis, we know that X is not in cell c_{i-1} at times $i-2$ or i. Only another Explorer of the same level could ever be in cell c_i, but by Theorem 3.28, no other Explorer ever explores that level.

We know that A remains in cell c_{i-1} for two rounds. If it is already moving back to the origin, it is not performing loneliness detection so clearly no *Search Phase* agents impact its behavior. If it is still moving west, all that matters is whether it detects being alone in either round in which it occupies cell c_{i-1}. By the above reasoning, *Search Phase* agents cannot cause it to detect that it is sharing its cell for both rounds in which it occupies cell c_{i-1}, proving the claim. □

Lemma 3.32 (Isolation Lemma 2)**.** *Let α' be a fixed execution that is identical to α except agents in the Allocation Phase treat agents in other phases as ghosts. Let T_E be the round in which A enters the Allocation Phase in α' and T_C the round in which it completes it. Let c'_i be the cell occupied by A at time i in α' for some i, $T_E \leq i \leq T_C$. Then we have $c_i = c'_i$.*

Proof. As in the proof of Isolation Lemma 1, we induct on i to prove our claim. First, we consider the case where $i = T_E$. By our definition of α', A appears at the origin in α' in the same round as it completed the *Separation Phase* in α, so we know $c_{T_E} = c'_{T_E}$.

If $i > T_E$, assume that we have $c_{i-1} = c'_{i-1}$. We prove that $c_i = c'_i$. To do so, we prove that the move chosen by A in round $i-1$ of α is not impacted by the presence of agents in other phases, and therefore it must be the same as the move chosen by A in round $i-1$ of α'.

By Lemmas 3.25 and 3.29, A never shares a cell with an agent in the *Search Phase*, so we only need to consider agents in the *Separation Phase*.

By Lemmas 3.4 and 3.25, we only need to consider the case where $c_{i-1} = (0, 0)$ or $(-1, 0)$. Since *Separation Phase* agents are only located at the origin at time 0, we know that A is never at the origin at the same time as an agent in the *Separation Phase*. If $c_{i-1} = (-1, 0)$, then A must have been executing *Part A* and reached the second to last move in the path in Figure 3-2. Since A does not detect loneliness in that cell, we know that no *Separation Phase* agent can influence the move chosen by A, proving our claim.

□

Lemma 3.33 (Isolation Lemma 3)**.** *Let α' be a fixed execution that is identical to α except agents in the Search Phase treat agents in other phases as ghosts. Let T be the round in which A enters the Search Phase in α'. Let c'_i be the cell occupied by A at time i in α' for some $i \geq T$. Then we have $c_i = c'_i$.*

Proof. As in the proofs of Isolation Lemma 1 and 2, we induct on i. Since the behavior of A is identical in α and α' up until A enters the *Search Phase*, we know if $i = T$, then $c_i = c'_i$.

If $i > T$, we assume that $c_{i-1} = c'_{i-1}$. We prove that $c_i = c'_i$. To do so, we prove that the move chosen by A in round $i - 1$ of α is not impacted by the presence of agents in other phases, and therefore it must be the same as the move chosen by A in round $i - 1$ of α'.

By Lemmas 3.25 and 3.29, A never shares a cell with an agent in the *Allocation* phase, so we only need to consider agents in the *Separation Phase*.

As noted in the previous proof, A only ever shares a cell with an agent in the *Separation Phase* agent if it is an Explorer with a status of *exploring*. With a status of *exploring*, A does not perform loneliness detection until after it has moved three times. After three moves, it is two moves above the west axis, and does not return to

a cell one move above the west axis with the same status of *exploring*. Therefore, we know that no agent in the *Separation Phase* can ever impact the behavior of A. $\square$

With these lemmas in mind, we can now prove that RECTANGLE-SEARCH locates the treasure eventually with probability 1 as long as $k \geq 19$.

Lemma 3.34. *Consider a probabilistic execution β of* RECTANGLE-SEARCH *where $k \geq 19$. Let L_i be the event that an agent locates the treasure at or before time i. Then we have*

$$\lim_{i \to \infty} P(L_i) = 1.$$

Proof. First, since all the agents begin at the origin at time 0, we know that the input assumptions of the *Separation Phase* are met by the initial configuration of the agents.

Combining Lemma 3.31 with Theorem 3.3, we have that the probability that at least 19 agents have completed the *Separation Phase* in β by some time T is at least

$$\left(1 - \left(1 - \frac{1}{2^{k-1}}\right)^{\left\lfloor \frac{T-6}{4} \right\rfloor}\right)^{19}.$$

Taking the limit, we have

$$\lim_{T \to \infty} \left(1 - \left(1 - \frac{1}{2^{k-1}}\right)^{\left\lfloor \frac{T-6}{4} \right\rfloor}\right)^{19} = 1,$$

so the probability that 19 agents complete the phase eventually in β is 1.

Now consider a *fixed* execution α in which at least 19 agents complete the *Separation Phase*. Note that unlike β, which is a probabilistic execution, α refers to some specfic non-probabilistic execution. We show that the treasure is located in α. Combining this with the previous reasoning proves the claim.

Combining Lemmas 3.31 and Theorem 3.1, we know that the first three input assumptions of the *Allocation Phase* are met in α. The fourth and final input assumption is given by our assumption that at least 19 agents complete the *Separation Phase* in α.

Because the input assumptions of the *Allocation Phase* are satisfied, we can apply Lemma 3.32 to see that the guarantees of the *Allocation Phase* are valid. These guarantees are given by Theorems 3.11, 3.12, 3.15, 3.19, 3.21, 3.22, 3.23, and 3.24.

Since the input assumptions of the *Search Phase* are met by the guarantees of the *Allocation Phase*, we know that Theorem 3.33 gives us that the guarantee of the *Search Phase* is met in α. This is given by Theorem 3.28.

If $D \leq 3$, then by Theorem 3.24 the agents in the *Allocation Phase* locate the treasure in α. By Theorem 3.28, if $D \geq 4$, the agents in the *Search Phase* locate the treasure in α, proving the claim.

□

Now that we know that RECTANGLE-SEARCH locates the treasure eventually with probability 1, we want to upper bound the number of rounds until it does so. Most of this work was already done in the *Analysis* sections for each of the phases (Sections 3.1.3, 3.2.3, and 3.3.3), but it remains to combine them to compute the expected number of rounds for the entire algorithm.

Before we can do that, we need to bound the number of Explorers that enter the *Search Phase*. In Lemma 3.35 we show that the number of Explorers who enter the phase before the treasure is located is upper bounded by a constant multiple of D. Recall from Section 3.2 we proved that the total number of explorers to enter the phase is a constant fraction of k (Theorem 3.16).

Lemma 3.35. *Consider some fixed execution α, and let i be the total number of Explorers that enter the Search Phase before the treasure is located in α. Then we have that $i \leq 5D + 8$.*

Proof. By Theorem 3.24, levels 0 through 3 are fully explored in the *Allocation Phase*. So, if $D+2 \leq 3$, then we know that the treasure will be located before any Explorers enter the *Search Phase*, so the lemma is true in that case.

We know that in the *Search Phase*, by the round in which some Explorer completes exploring level d, levels 4 through $d-1$ have already been explored. So, by the round in which level $D+2$ is explored, levels 0 through $D+1$ have already been explored. Thus, we just need to upper bound the number of Explorers that could have entered the phase before level $D+2$ was completely explored.

Even if each Explorer only explored a single level, the $D/2^{th}$ Explorer would end up exploring level $D+2$. Let X be the Explorer that eventually explores level $D+2$. We just need to compute the number of Explorers that could enter the phase after X, before it completes exploring level $D+2$.

Since by Input Assumption 2, all Explorers enter the phase in level 4, we know that it takes $D-2$ rounds for X to reach level $D+2$. Now, by Lemma 3.27, we know that it will take X $8(D+2)+5 = 8D+21$ rounds before all level $D+2$ cells have been searched. So, there are $9D+19$ rounds between when X enters the phase and when it completes exploring level $D+2$.

By Input Assumption 7, Explorers only enter the phase in even-numbered rounds, so at most $4D/9+9$ agents enter the phase during these rounds. Now, since $D/2-1$ Explorers entered the phase before X, this leads to a total of $5D+8$ Explorers that could have entered the phase before the treasure was located.

□

Finally, we have all the tools we need to upper bound the expected number of rounds until RECTANGLE-SEARCH locates the treasure.

In Lemma 3.35 that we proved that the number of Explorers to enter the *Search Phase* before the treasure is located is at most $5D+8$. In Lemma 3.16, we proved that at least $\lfloor (k-9)/5 \rfloor$ agents become Explorers.

In Lemma 3.36, we upper bound the expected number of rounds until RECTANGLE-SEARCH locates the treasure if $\lfloor (k-9)/5 \rfloor > 5D+8$. Then in Lemma 3.37 we upper bound the expected number of rounds in the other case, where $\lfloor (k-9)/5 \rfloor \leq 5D+8$. Then we combine these results in Theorem 3.38.

Lemma 3.36. *Consider a probabilistic execution β of* RECTANGLE-SEARCH *in which $\lfloor (k-9)/5 \rfloor > 5D+8$. Some agent locates the treasure in β in $f_k(O(D))+O(1)$ rounds in expectation.*

Proof. Let i be a random variable representing the number of Explorers that enter the *Search Phase* before the treasure is located. By Lemma 3.35, $i \leq 5D+8$. Thus, we must have $i < \lfloor (k-9)/5 \rfloor$. From Lemma 3.16, we know that at least $\lfloor (k-9)/5 \rfloor$ Explorers eventually enter the phase. Thus, there is at least one Explorer which does not complete the *Allocation Phase* by the round in which some agent locates the treasure.

Let X_{i+1} be a random variable representing the $i+1^{th}$ Explorer to enter the *Search Phase* in any particular branch of β. By our definition of i, X_{i+1} does not complete the *Allocation Phase* by the round in which the treasure is located. So, to upper bound the expected number of rounds until the treasure is found we can upper bound the expected number of rounds until X_{i+1} completes the *Allocation Phase*.

We compute the expected number of rounds until X_{i+1} completes the *Allocation Phase* conditioned on X_{i+1} executing *Part A*, and on it executing *Part B*. We will see that in either case, the expected number of rounds is at most $f_k(O(D)) + O(1)$, so then we can use the law of Total Expectation to conclude that the expected number of rounds is always at most $f_k(O(D)) + O(1)$.

We first consider the case where X_{i+1} executes *Part B* of the *Allocation Phase*, because we will use this reasoning in the case where X_{i+1} executes *Part A*. By Lemma 3.14, at most 14 agents execute *Part A* of the *Allocation Phase*, and by Lemma 3.13, X_{i+1} is the $5(i+1)^{th}$ agent to begin executing *Part B*. So we have that

X_{i+1} must have been at most the $14 + 5(i+1)^{th}$ agent to complete the *Separation Phase*.

By the definition of f_k, the expected number of rounds until X_{i+1} completes the *Separation Phase* is at most $f_k(14 + 5(i+1))$. By Lemma 3.26, X_{i+1} only spends $O(1)$ rounds in the *Allocation Phase*, so the expected number of rounds before X_{i+1} completes the *Allocation Phase* is $f_k(14+5(i+1)) + O(1)$. By Lemma 3.35, we have $i = O(D)$, so this simplifies to $f_k(O(D)) + O(1)$.

If X_{i+1} executes *Part A* of the *Allocation Phase*, then by Lemma 3.18 we know that it completes the *Allocation Phase* exactly two rounds after the first *Part B* Explorer. By the above reasoning, the first *Part B* Explorer is at most the 19^{th} agent to complete the *Separation Phase*, so it does so by round $f_k(19)$ in expectation. So then X_{i+1} completes the phase by round $f_k(19) + 2$ in expectation.

The expected number of rounds before X_{i+1} completes the *Allocation Phase* is $f_k(O(D)) + O(1)$ if X_{i+1} executes *Part B*, and $f_k(19) + 2 \leq f_k(O(D)) + O(1)$ if X_{i+1} executes *Part A*. Applying the law of Total Expectation, we know that the expected number of rounds until X_{i+1} completes the *Allocation Phase* is $f_k(O(D)) + O(1)$.

□

Lemma 3.37. *Consider a probabilistic execution β of* RECTANGLE-SEARCH *in which $\lfloor (k-9)/5 \rfloor \leq 5D + 8$. Some agent locates the treasure in β in $f_k(O(D)) + O(D^2/k)$ rounds in expectation.*

Proof. Clearly $\lfloor (k-9)/5 \rfloor \leq 5D + 8$ implies $k = O(D)$. Since k is small relative to D in this case, we will upper bound the expected number of rounds until all agents complete the *Separation Phase*, and then analyze the number of rounds after that until the treasure is located.

Let X_j be a random variable representing the last Explorer to enter the *Search Phase* in any particular branch of β. Since X_j clearly completes the *Separation Phase* by the round in which the last agent does so, we know that X_j completes the

Separation Phase by the end of round $f_k(k)$ in expectation.

By Lemma 3.26, if X_j executes *Part B* of the *Allocation Phase*, it spends $O(1)$ rounds in that phase, so in expectation, after $f_k(k) + O(1)$ rounds, X_j enters the *Search Phase*. By the same reasoning as in Lemma 3.36, if X_j executes *Part A* of the *Allocation Phase*, it completes the *Allocation Phase* in $f_k(19) + 2$ rounds in expectation. As in the previous lemma, we can apply the law of Total Expectation to conclude that the expected number of rounds until X_j completes the *Allocation Phase* is $f_k(k) + O(1)$.

By Theorem 3.24, after X_j completes the *Allocation Phase*, levels 0 through 3 have been completely searched. Applying Lemma 3.30 with $i = j$ and $d = D + 2$, we have that levels 4 through $D + 2$ are searched once X_j has been in the *Search Phase* for $O(D + D^2/j)$ rounds. Combining this with the previous reasoning, we have that the expected number of rounds until levels 0 through $D + 2$ are searched completely is $f_k(k) + O(D + D^2/j)$.

Using the facts that $k = O(D)$ and $j = \Theta(k)$ (Theorem 3.16), this simplifies to $f_k(O(D)) + O(D^2/k)$, concluding this proof. □

Theorem 3.38. *Consider a probabilistic execution β of* Rectangle-Search. *Some agent locates the treasure in β in $f_k(O(D)) + O(D^2/k)$ rounds in expectation.*

Proof. By Lemma 3.36, the expected number of rounds until the treasure is located if $\lfloor (k-9)/5 \rfloor > 5D+8$ is $f_k(O(D))+O(1)$. If $\lfloor (k-9)/5 \rfloor \leq 5D+8$, then by Lemma 3.37 the expected number of rounds until the treasure is located is $f_k(O(D)) + O(D^2/k)$.

In either case, the expected number of rounds until the treasure is located is at most $f_k(O(D)) + O(D^2/k)$, so we can conclude that the expected number of rounds until some agent locates the treasure in β is $f_k(O(D)) + O(D^2/k)$. □

Now, we can apply our upper bound for f_k from Section 3.1.3 to upper bound the expected runtime of Rectangle-Search.

Theorem 3.39. *Consider a probabilistic execution β of* Rectangle-Search. *Some agent locates the treasure in β in $O(D \cdot \log k + D^2/k)$ rounds in expectation.*

Proof. This is the result of applying Lemma 3.9 to Lemma 3.38. □

Recall in Section 3.1.3 that we conjectured an even tighter bound for f_k. Applying this tighter bound results in another conjecture specifying a tighter bound for Rectangle-Search.

Conjecture 3.40. *Consider a probabilistic execution β of* Rectangle-Search. *Some agent locates the treasure in β in $O(D \cdot \log\log k + D^2/k + \log k)$ rounds in expectation.*

Intuition. This is the result of applying Conjecture 3.10 to Lemma 3.38.

Chapter 4

Improving the Runtime

In this chapter, we will describe an algorithm Hybrid-Search. This algorithm locates the treasure in $O(D \cdot \log k + D^2/k)$ rounds in expectation. Furthermore, we conjecture that it locates the treasure in $O(D \cdot \log\log k + D^2/k)$ rounds in expectation. Like in [2], Hybrid-Search combines an algorithm Rectangle-Search, which locates the treasure efficiently when D is large relative to k, with an algorithm Geometric-Search, which locates the treasure efficiently when D is small relative to k.

We will note that while Rectangle-Search alone locates the treasure in $O(D \cdot \log k + D^2/k)$ rounds in expectation ($O(D \cdot \log\log k + D^2/k + \log k)$ by Conjecture 3.40), it is guaranteed to locate the treasure eventually as long as $k \geq 19$. As we will see, Hybrid-Search only finds the treasure at all with high probability. So, it is possible (although unlikely), that an execution of Hybrid-Search could fail to locate the treasure.

4.1 The Geometric-Search Algorithm

Geometric-Search is a simple algorithm which locates the treasure with high probability when $D \leq (\log k)/2$. To execute Geometric-Search, an agent uses

$\delta = \delta$-GEOMETRIC and $M = M$-GEOMETRIC, with the states and initial values defined in the following section.

4.1.1 The Algorithm

Informal Description

The agent begins by randomly choosing a quadrant to explore. Without loss of generality, assume that the agent chooses the north-east quadrant. The agent begins by moving east once. In the next round, with probability 1/2 it moves east again, and with probability 1/2 it moves north. It repeats this process as long as it continues moving east. Once it moves north once, it moves north in every future round.

Formal Description

Each state $s \in S$ contains the following variables. The variables that are not used in this algorithm are omitted here.

quad: member of $\{1, 2, 3, 4, \text{NONE}\}$, initially NONE
coin: member of $\{\text{HEADS}, \text{TAILS}\}$, initially HEADS

δ-GEOMETRIC$(s, alone, origin)$

if $s.quad =$ NONE
    $coin1 :=$ result of fair coin toss
    $coin2 :=$ result of fair coin toss
    if $coin1 =$ HEADS and $coin2 =$ HEADS
        $s.quad := 1$
    elseif $coin1 =$ HEADS and $coin2 =$ TAILS
        $s.quad := 2$
    elseif $coin1 =$ TAILS and $coin2 =$ HEADS
        $s.quad := 3$
    elseif $coin1 =$ TAILS and $coin2 =$ TAILS
        $s.quad := 4$
elseif $s.coin =$ HEADS
    $s.coin :=$ result of fair coin toss
return s

M-GEOMETRIC(s)
  if ($s.quad = 1$ and $s.coin =$ TAILS) or ($s.quad = 2$ and $s.coin =$ HEADS)
    return NORTH
  elseif ($s.quad = 1$ and $s.coin =$ HEADS) or ($s.quad = 4$ and $s.coin =$ TAILS)
    return EAST
  elseif ($s.quad = 3$ and $s.coin =$ TAILS) or ($s.quad = 4$ and $s.coin =$ HEADS)
    return SOUTH
  elseif ($s.quad = 3$ and $s.coin =$ HEADS) or ($s.quad = 2$ and $s.coin =$ TAILS)
    return WEST
  return NONE

4.1.2 Correctness

We begin by showing that if the treasure is located close to the origin, then it is very likely that some agent executing GEOMETRIC-SEARCH will locate it. Before presenting the proof, we introduce a Chernoff bound that will be useful through this chapter. It states that for a Bernoulli random variable X, for any $\epsilon > 0$ we have

$$P\left(X < (1-\epsilon) \cdot E[X]\right) \leq \frac{1}{e^{(\epsilon^2 \cdot E[X])/2}}.$$

With this in mind, we present the following lemma.

Lemma 4.1. *Consider a probabilistic execution β of* GEOMETRIC-SEARCH *where $k \geq 19$. If $D \leq (\log k)/2$, then the probability that no agent locates the treasure in β is at most $1/e^{\sqrt{k}/8-1}$.*

Proof. Say that the treasure is located in some cell c with coordinates (x, y) for some $x > 0$, $y \geq 0$. Some agent A which is exploring the north-east quadrant will begin by moving east once. It will explore cell c if it moves east for $x-1$ more rounds and then moves north. So, the probability that A will explore cell c, given that it is exploring the north-east quadrant, is $1/2^x$.

Let n be the number of agents exploring the north-east quadrant. Let F be the

event that no agent explores c. Then we have

$$P(F) = \left(1 - \frac{1}{2^x}\right)^n \leq \frac{1}{e^{(n/2^x)}}.$$

In expectation, $k/4$ agents will explore each quadrant. We will show that the probability that very few agents explore any given quadrant is small. Applying the Chernoff bound from above with $\epsilon = 1/2$ yields

$$P(n < k/8) \leq \frac{1}{e^{k/32}}.$$

Now we can use the law of Total Probability to upper bound the probability that no agent explores c.

$$\begin{aligned} P(F) &= P\left(F \,\middle|\, n \geq \frac{k}{8}\right) \cdot P\left(n \geq \frac{k}{8}\right) + P\left(F \,\middle|\, n < \frac{k}{8}\right) \cdot P\left(n < \frac{k}{8}\right) \\ &\leq \frac{1}{e^{k/(8 \cdot 2^x)}} \cdot 1 + 1 \cdot \frac{1}{e^{k/32}} \\ &\leq \frac{2}{e^{k/(8 \cdot 2^x)}} \\ &\leq \frac{1}{e^{k/(8 \cdot 2^x) - 1}}. \end{aligned}$$

We know that $0 < x \leq (\log k)/2$. Since this probability is maximized when $x = (\log k)/2$, for all cells c at most $(\log k)/2$ steps from the origin we have

$$P(F) \leq \frac{1}{e^{k/(8 \cdot 2^{\log k/2}) - 1}} = \frac{1}{e^{\sqrt{k}/8 - 1}},$$

proving the claim. □

4.1.3 Analysis

In this section, we show that if an agent executing Geometric-Search did locate the treasure, then it must have done so very efficiently.

Lemma 4.2. *Let α be a fixed execution of* GEOMETRIC-SEARCH *in which some agent A locates the treasure. Then A locates the treasure in $O(D)$ rounds.*

Proof. Say that the treasure is located in some cell (x, y). Without loss of generality assume that A was exploring the north-east quadrant. To have located the treasure, A must have moved east x times followed by north y times, and therefore located the treasure in exactly D rounds. □

4.2 The Hybrid-Search Algorithm

Finally, we present the algorithm HYBRID-SEARCH, which combines the strengths of both RECTANGLE-SEARCH and GEOMETRIC-SEARCH. To execute HYBRID-SEARCH, an agent uses $\delta = \delta$-HYBRID and $M = M$-HYBRID, with the states and initial values as defined below.

4.2.1 The Algorithm

Informal Description

The agent begins by choosing whether to execute RECTANGLE-SEARCH or GEOMETRIC-SEARCH, each with probability 1/2. If it chooses GEOMETRIC-SEARCH, it immediately begins executing GEOMETRIC-SEARCH. If it chooses RECTANGLE-SEARCH, it pauses for one round, and then begins executing RECTANGLE-SEARCH.

Formal Description

Each state $s \in S$ contains the following variables. The variables that are not used directly in the following functions are omitted here.

algorithm: member of {GEOMETRIC, RECTANGLE, NONE}, initially NONE
phase: member of {SEPARATION, ALLOCATION, SEARCH, NONE}, initially NONE
pause: boolean, initially FALSE

```
δ-HYBRID(s, alone, origin)
    if s.algorithm = NONE
        coin := result of fair coin toss
        if coin = HEADS
            s.algorithm = RECTANGLE
            s.phase = SEPARATION
            Initialize other state variables for Separation Phase
            s.pause = TRUE
        elseif coin = TAILS
            s.algorithm = GEOMETRIC
            Initialize other state variables for GEOMETRIC-SEARCH
    elseif s.algorithm = RECTANGLE
        if s.pause
            s.pause := FALSE
        else
            s := δ-RECTANGLE(s, alone, origin)
    if s.algorithm = GEOMETRIC
        s := δ-GEOMETRIC(s, alone, origin)
    return s

M-HYBRID(s)
    if s.algorithm = RECTANGLE and not s.pause
        return M-RECTANGLE(s)
    elseif s.algorithm = GEOMETRIC
        return M-GEOMETRIC(s)
    return NONE
```

4.2.2 Correctness

In round 1, agents randomly decide which algorithm to execute. We have already proved the correctness of RECTANGLE-SEARCH and GEOMETRIC-SEARCH. So, to prove the correctness of HYBRID-SEARCH, we just need to prove that the agents executing the two algorithms will not interfere with each other.

Lemma 4.3. *Let α be a fixed execution of* HYBRID-SEARCH *in which at least one agent executes each of* RECTANGLE-SEARCH *and* GEOMETRIC-SEARCH. *Let A be an agent that chooses to execute* RECTANGLE-SEARCH *and A' an agent that chooses to execute* GEOMETRIC-SEARCH. *Then after round* 1, *A and A' never share a cell.*

Proof. In round 2, A' moves in a randomly chosen direction, and A remains at the origin (because RECTANGLE-SEARCH agents pause once before beginning to execute RECTANGLE-SEARCH). In round 3, A will begin executing RECTANGLE-SEARCH.

A' moves further from the origin every round. Since it begins moving in the round before A, and A can only move once per round, A will always be closer to the origin than A', and thus will never share its cell. □

As we have mentioned, there are some executions of HYBRID-SEARCH in which no agent ever locates the treasure. In the following lemma, we upper bound the probability that no agent executing HYBRID-SEARCH ever locates the treasure.

First, we introduce a definition. If $D \leq \log(k/3)/2$, we say that an execution of HYRBID-SEARCH is *correct* if an agent executing GEOMETRIC-SEARCH locates the treasure. If $D > \log(k/3)/2$, we say that an execution of HYBRID-SEARCH is *correct* if an agent executing RECTANGLE-SEARCH locates the treasure. Clearly if an execution of HYBRID-SEARCH is correct, then some agent locates the treasure. Thus, we will upper bound the probability that HYBRID-SEARCH is not correct, which will upper bound the probability that no agent executing HYBRID-SEARCH locates the treasure.

Lemma 4.4. *Consider a probabilistic execution β of* HYBRID-SEARCH *where $k \geq 57$. The probability that β is not correct is at most $1/e^{\sqrt{k/3}/8-2}$.*

Proof. First we consider the case where $D > \log(k/3)/2$. Let n_R be the number of agents that choose to execute RECTANGLE-SEARCH in β. If $n_R \geq 19$, then by Lemma 3.34 the probability that some agent locates the treasure eventually is 1. So, we want to upper bound the probability that $n_R < 19 \leq k/3$.

Note that $E[n_R] = k/2$. So then applying our Chernoff bound with $\epsilon = 1/3$ yields

$$P(n_R < 19) \leq P(n_R < k/3) \leq \frac{1}{e^{k/36}}.$$

So then we know that if $D > \log(k/3)/2$, the probability that β is not correct is at most $1/e^{k/36}$.

Now we consider the case where $D \leq \log(k/3)/2$. Let F be the event that no agent executing GEOMETRIC-SEARCH locates the treasure in β, and let n_G be the number of agents that execute GEOMETRIC-SEARCH. If $n_G \geq k/3$, then we can apply Lemma 4.1. Then, we show that it is very unlikely that $n_G < k/3$, and combine the results using the law of Total Probability.

$$\begin{aligned} P(F) &= P\left(F \,\middle|\, n_G \geq \frac{k}{3}\right) \cdot P\left(n_G \geq \frac{k}{3}\right) + P\left(F \,\middle|\, n_G < \frac{k}{3}\right) \cdot P\left(n_G < \frac{k}{3}\right) \\ &\leq \frac{1}{e^{\sqrt{k/3}/8-1}} \cdot 1 + 1 \cdot \frac{1}{e^{k/36}} \\ &\leq \frac{1}{e^{\sqrt{k/3}/8-2}}. \end{aligned}$$

Our bound is tighter for $D > \log(k/3)/2$, so we can conclude that in general, the probability that HYRBID-SEARCH is not correct is at most $1/e^{\sqrt{k/3}/8-2}$, as desired.

□

4.2.3 Analysis

Now that we have assessed the likelihood that HYBRID-SEARCH will locate the treasure, it remains to upper bound the expected number of rounds until it does so.

Theorem 4.5. *Consider a probabilistic execution β of* HYBRID-SEARCH *with $k \geq 57$ which is correct. Then the expected number of rounds until some agent locates the treasure in β is $O(D)$ if $D \leq \log(k/3)/2$, and $f_k(O(D)) + O(D^2/k)$ otherwise.*

Proof. First consider the case where $D \leq \log(k/3)/2$. By our assumption that β is correct, we know that in this case an agent executing GEOMETRIC-SEARCH will locate the treasure. By Lemma 4.2, this takes $O(D)$ rounds.

Now consider the case where $D > \log(k/3)/2$. Let T be a random variable

representing the round in which the treasure is located in β. Let n_R be the number of agents that execute Rectangle-Search. If $n_R \geq k/3$, then by Theorem 3.38 we know that the expected number of rounds until some agent locates the treasure is at most $f_k(O(D)) + O(D^2/k)$. We know that $n_R \geq 1$, so even when $n_R < k/3$, then the expected number of rounds until some agent locates the treasure is at most $f_k(O(D)) + O(D^2)$. We can combine these cases using the law of Total Probability.

$$\begin{aligned} E[T] &\leq E\left[T \,\middle|\, n_R \geq \frac{k}{3}\right] \cdot P\left(n_R \geq \frac{k}{3}\right) + E\left[T \,\middle|\, n_R < \frac{k}{3}\right] \cdot P\left(n_R < \frac{k}{3}\right) \\ &\leq \left(f_k(O(D)) + O(D^2/k)\right) \cdot 1 + \left(f_k(O(D)) + O(D^2)\right) \cdot \frac{1}{e^{k/36}} \\ &\leq f_k(O(D)) + O(D^2/k), \end{aligned}$$

as desired. □

If $f_k(i) = \Theta(i \cdot \log k)$, then Rectangle-Search locates the treasure in the same asymptotic number of rounds as Hybrid-Search. However, if, as we suggest in Conjecture 3.10, $f_k(i) = O(\log k + i \cdot \log\log k)$, then Hybrid-Search locates the treasure in $O(D \cdot \log\log k + D^2/k)$ rounds in expectation, which is faster than the $O(D \cdot \log\log k + D^2/k + \log k)$ rounds it would take Rectangle-Search in expectation. We note that if $f_k(D) = O(\log k + i)$, then Hybrid-Search locates the treasure in the optimal $O(D + D^2/k)$ rounds in expectation. While we do not believe this is the case, we have no proof otherwise.

Chapter 5

Conclusion

In some sense, loneliness detection is the minimal form of communication, because agents only exchange a single bit of information. In this paper, we have shown that with this small amount of communication, agents are able to locate the treasure in $O(D \cdot \log k + D^2/k)$ rounds in expectation. This is an improvement upon the lower bound of $\Omega((D + D^2/k) \cdot \log^{1+\epsilon} k)$ for some $\epsilon > 0$ which is required of agents which have no ability to communicate and no knowledge of k [1]. In fact, this lower bound applies to agents with unlimited memory. For our algorithms, we only require that our agents possess constant memory.

In this paper we presented two algorithms which locate the treasure in $O(D \cdot \log k + D^2/k)$ rounds in expectation: Rectangle-Search and Hybrid-Search, which uses Rectangle-Search as a subroutine. The benefit of using Hybrid-Search is only clear when considered with Conjecture 3.10. If our conjecture is correct, then Rectangle-Search locates the treasure in $O(D \cdot \log \log k + D^2/k + \log k)$ rounds in expectation, while Hybrid-Search locates it in only $O(D \cdot \log \log k + D^2/k)$ rounds in expectation.

Future work should focus on developing a tight bound for the *Separation* completion function f_k. This could begin with a proof of Conjecture 3.10, which states

that $f_k(i) = O(\log k + i \cdot \log \log k)$. An even better outcome would be to prove that $f_k(i) = O(\log k + i)$, which would prove that HYBRID-SEARCH is an optimal algorithm.

If HYBRID-SEARCH cannot be shown to locate the treasure in $O(D+D^2/k)$ rounds in expectation, then it is still an open question as to whether it is possible to locate the treasure in $O(D + D^2/k)$ rounds in this model. Future work may focus on developing a different algorithm for locating the treasure which meets this bound, or proving a tighter lower bound on the number of rounds required to locate the treasure in this model.

Finally, it may be interesting to study whether HYBRID-SEARCH can be modified to work in more general models. For example, in [2], Emek et al. show how to modify their algorithm to work in the case where the agents move asynchronously. In [5], Langner et al. show how to modify the algorithm from [2] to work in the event of failures. However, we do note that the simple error-handling strategy of duplicating each agent will not work in this model, because duplicates would interfere with loneliness detection.

Bibliography

[1] Ofer Feinerman, Amos Korman, Zvi Lotker, and Jean-Sebastien Sereni. Collaborative search on the plane without communication. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, 2012.

[2] 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*, 2014.

[3] 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 ACM Symposium on Principles of Distributed Computing*, 2014.

[4] Deborah M Gordon. *Ant encounters: interaction networks and colony behavior*. Princeton University Press, 2010.

[5] Tobias Langner, David Stolz, Jara Uitto, and Roger Wattenhofer. Fault-Tolerant ANTS. In *Proceedings of the 28th International Symposium on Distributed Computing*, 2014.

[6] Mohsen Ghaffari, Nancy Lynch, Srinkanth Sastry. Leader Election Using Loneliness Detection. *Distributed Computing*, 25(6): 427-450, 2012. Special issue for *DISC 2011*.