How Many Ants Does It Take To Find the Food?

Yuval Emek[a], Tobias Langner[b], David Stolz[b,*], Jara Uitto[b], Roger Wattenhofer[b]

[a] *Technion, Israel*
[b] *ETH Zürich, Switzerland*

Abstract

Consider the *Ants Nearby Treasure Search (ANTS)* problem, where n mobile agents, initially placed at the origin of an infinite grid, collaboratively search for an adversarially hidden treasure. The agents are controlled by deterministic/randomized finite or pushdown automata and are able to communicate with each other through constant-size messages. We show that the minimum number of agents required to solve the ANTS problem crucially depends on the computational capabilities of the agents as well as the timing parameters of the execution environment. We give lower and upper bounds for different scenarios.

Keywords: treasure search, finite automata, collaborative search

1. Introduction

Recent research on understanding the behavior of insect colonies from a distributed computing perspective has mainly focused on questions like "How long does it take a large collection of ants to locate a food source?" [1, 2] or "How do the computational capabilities of a single ant within this collection affect the time until the food source is found?" [3, 4, 5, 6].

In this paper, we take a computability point of view and, instead of focusing on *large* numbers of agents and on the time required to find a food source, analyze the *minimum* number of agents that is required to locate a food source within (expected) finite time. More precisely, we show that the minimally required number of agents crucially depends on the model assumptions, e.g., whether the environment is synchronous or asynchronous.

While different models of synchronization have been studied earlier in the literature, we take a look at two different aspects that have crucial effects on the difficulty of our search problem. On one hand, we study both randomized

*Corresponding author. Address: Gloriastrasse 35, ETZ G63, 8092 Zürich, Switzerland. Phone: +41 44 63 26894

Email addresses: `yemek@ie.technion.ac.il` (Yuval Emek), `tobias.langner@tik.ee.ethz.ch` (Tobias Langner), `stolzda@ethz.ch` (David Stolz), `jara.uitto@tik.ee.ethz.ch` (Jara Uitto), `wattenhofer@ethz.ch` (Roger Wattenhofer)

and deterministic protocols in contrast to the aforementioned previous work, that focused only on the randomized case. Our results indicate that the search problem becomes strictly harder in the deterministic setting.

On the other hand, we extend the previous work with respect to the choice of the model of computation. While previous work considered agents that are controlled by *finite automata* (FA) or *Turing machines*, we study a model that is, at least according to Chomsky's hierarchy, inbetween these two models, namely a *pushdown automaton* (PDA). While it might intuitively feel that providing the agent(s) with infinite memory makes the problem trivial, we show that in the determistic model, this is not necessarily the case.

For all combinations of the aforementioned characteristics, we establish lower and upper bounds on the number of agents required to locate the food. Our bounds are tight in most cases. We note the striking resemblance to the problem of finding the number of people needed to change a light bulb: For people, the answer usually depends on nationality and profession while for ants, it depends on timing and computational power.

We essentially present two different families of algorithms – rectangle/spiral searches and geometric searches – which are inspired by results of Emek et al. [1]. The main contributions of this paper, however, are the lower bounds for two deterministic FA- and one deterministic PDA-agent presented in Section 4.1 and Section 5.2, respectively. Table 1 gives a complete picture of our findings.

As border cases of our findings, we point out that in an asynchronous setting four agents are sufficient to solve the problem when their computational capabilities are most restricted, i.e., they are controlled by deterministic FAs. If we allow access to random bits and grant the agents slightly more computational power – a PDA – already one single agent can solve the problem. Note that neither of these results require the full computational power of a Turing machine.

We do not claim that our considerations are particularly relevant from a biological perspective – an ant hive generally consists of significantly more than four ants. However, our results show that powerful computational capabilities can be traded for primitive means of communication while still being able to solve complex problems – even for small number of agents.

1.1. Related Work

Our work is inspired by Feinerman et al. who proposed a problem called *ants nearby treasure search (ANTS)*, where n *ants*, or *agents*, are searching the plane [2, 3]. The agents are controlled by Turing machines and are not allowed to communicate with each other after leaving the origin. Assuming a knowledge of a constant approximation of n, the agents are able to locate the treasure in time $\mathcal{O}(D + D^2/n)$ where D is the distance to the treasure. Furthermore, Feinerman et al. observe a matching lower bound and prove that this lower bound cannot be matched without some knowledge of n.

There are two fundamental differences between the model studied by Feinerman et al. and our models. First, our agents are operated by finite automata or pushdown automata. The stronger computational model provided by Turing machines enables individual agents to accomplish tasks way beyond our capabilities, such as performing spiral searches and remembering the execution history. In a recent related work, Lenzen et al. study the effects that bounding the memory of the agents and the range of available probabilities have on the runtime [5]. In particular, they show that the time needed to discover can be expressed as a function of D, n and ℓ, where $\ell \in \mathcal{O}(\log D)$ and an agent can use at most $\mathcal{O}(\ell)$ random bits per round.

Second, our agents are allowed to communicate outside the origin, yet only through constant-size messages – a model which was also studied by Emek et al. [1]. Another previously studied way to describe such communication is the pheromone model, where the agents are allowed to leave marks into the grid that the other agents can sense [7]. In our model, the agents are allowed to communicate with other agents in the same cell, which is, to some extend, analogous to the agents bumping into each other.

The general concept of graph exploration is widely studied in computer science. Typically, given a graph, the task is to visit all nodes by walking along the edges [8, 9, 10, 11, 12]. It is well-known that random walks allow a single agent to visit all nodes of a finite undirected graph in expected polynomial time [13]. Note that there are infinite graphs, such as a grid, where the expected time for a random walk to reach any designated node is infinite. Our problem can also be seen as a variant of the game of cops and robbers, where the robber remains dormant [14].

The classic example of a treasure finding problem is the *cow-path* problem. The task in the cow-path problem is to find a treasure on a line as quickly as possible. This task can be solved with a constant competitive ratio with a deterministic algorithm. The optimal algorithm for the 2-dimensional version is a simple spiral search [15]. The problem has also been studied in a multi-agent setting by López-Ortiz and Sweet [16].

Also finite automata searching a graph have been studied earlier [4]. Other work considering distributed computing by finite automata includes for example *population protocols* [17, 18]. Recently, a new general model of computation in graphs was introduced, where the nodes are controlled by finite automata instead of Turing machines [19]. The main connection to our work is that we use an equivalent communication model.

1.2. Model

We consider a variant of [2]'s ANTS problem, where a set of mobile *agents* search the infinite grid for an adversarially hidden treasure. Our model is an adapted version of the model used in a paper by Emek et al. [1]. Each agent is controlled either by a finite automaton or by a pushdown automaton, both either deterministic or randomized, with a common sense of direction and can communicate only with agents sharing the same grid cell.

More formally, consider n mobile agents that explore $\mathbb{Z}^2$. In the beginning of the execution, all agents are positioned in the same grid cell referred to as the *origin* (say, the cell with coordinates $(0,0) \in \mathbb{Z}^2$). In contrast to prior work, we do *not* assume that the agents can distinguish between the origin and the other cells (see Section 1.5 for a discussion of this matter). We refer to the cells with either x or y-coordinate equal to 0 as *north/east/south/west-axis*, depending on which part of the plane they are located. For example, the north-axis corresponds to the set of cells $\{(0,y) \mid y \geq 0\}$.

The *distance* $\mathrm{dist}(c,c')$ between two grid cells $c = (x,y)$ and $c' = (x',y')$ in $\mathbb{Z}^2$ is defined with respect to the ℓ_1 norm (a.k.a. Manhattan distance), that is, $|x-x'| + |y-y'|$. Two cells are called *neighbors* if the distance between them is 1. In each step of the execution, agent a positioned in cell $(x,y) \in \mathbb{Z}^2$ can either move to one of the four neighboring cells $(x,y+1), (x,y-1), (x+1,y), (x-1,y)$, or stay put in cell (x,y). The former four *position transitions* are denoted by the corresponding cardinal directions N, E, S, W, whereas the latter (stationary) position transition is denoted by P (standing for "stay put"). We point out that the agents have a common sense of orientation, i.e., the cardinal directions are aligned with the corresponding grid axes for every agent in every cell.

In an *asynchronous environment*, each agent's execution progresses in discrete (asynchronous) steps indexed by the non-negative integers. We denote the time at which agent a completes step $i > 0$ by $t_a(i) > 0$ and call $t_a(i)$ an *activation time*. Following common practice, we assume that the activation times $t_a(i)$ for all a and i are determined by the policy ψ of an adversary that knows the protocol but is oblivious to its random bits, whereas the agents do not have any sense of time. The set of activation times determined by the adversary is called a *schedule* and thus, the policy of an adversary is a function from the set of possible protocols to the set of possible schedules, which is independent of the random coin tosses. Throughout the paper, we will use the terms synchronous/asynchronous policy and -/- schedule interchangeably in the rest of the paper, despite their subtle difference. A *synchronous environment* corresponds to the special case where $t_a(i) = i$ for all agents a and all $i > 0$.

The communication and computational capabilities of the agents are limited. Specifically, in our model, an agent a positioned in cell $c \in \mathbb{Z}^2$ can communicate with all other agents positioned in cell c at the same time. This communication is limited though: agent a merely senses for each state q of its (finite or pushdown) automaton, whether there exists at least one agent $a' \neq a$ in cell c whose current state is q. Notice that this communication scheme is a special case of the one-two-many communication scheme introduced in [19] with bounding parameter $b = 1$.

Since we only consider instances with a constant number of agents, we allow each agent to run a different individual protocol. This is modeled by assigning to each agent an individual initial state in the respective automaton (note that this is only relevant in the deterministic case as otherwise coin flips can be used to separate agents). The protocol is controlled by either a finite automaton or a pushdown automaton. We shall first explain the semantics of the former and then explain the additional capabilities of the latter.

FA-protocol. When an agent employs an *FA-protocol*, it has a constant memory and thus, in general, cannot store coordinates in $\mathbb{Z}^2$. Formally, the agent's protocol is captured by the 3-tuple $\Pi = \langle Q, s_0^a, \delta \rangle$, where Q is the finite set of *states*, $s_0^a \in Q$ is the *initial state* of agent a, and $\delta : Q \times 2^Q \to 2^{Q \times \{N,S,E,W,P\}}$ is the *transition function*. To allow the agents to perform different tasks also in the absence of randomization, each agent a has a unique start state s_0^a in which it resides at time 0. Suppose that at time $t_a(i)$, agent a is in state $q \in Q$ and positioned in cell $c \in \mathbb{Z}^2$. Then, the state $q' \in Q$ of agent a at time $t_a(i+1)$ and its corresponding movement $\tau \in \{N, S, E, W, P\}$ are dictated based on the transition function δ by picking the tuple (q', τ) uniformly at random from $\delta(q, Q_a)$, where $Q_a \subseteq Q$ contains state $p \in Q$ if and only if there exists some (at least one) agent $a' \neq a$ such that a' is in state p and positioned in cell c at time $t_a(i)$. A FA-protocol is *deterministic* if each step is deterministic, i.e., $|\delta(q, Q_a)| \leq 1$ for all $q \in Q$ and $Q_a \subseteq Q$. For simplicity, we assume that while Q_a (input to δ) is determined based on the status of cell c at time $t_a(i)$, the actual application of the transition function δ occurs instantaneously at the end of the step, i.e., agent a is considered to be in state q and positioned in cell c throughout the time interval $[t_a(i), t_a(i+1))$.

PDA-protocol. When an agent employs a *PDA-protocol*, it is controlled by a pushdown automaton with an infinite stack. The communication and movement model remains the same. The only addition is that in each step, an agent reads and removes the top-most symbol from the stack ("pop") – if the stack is empty, the agent reads the special symbol ε and the stack remains unchanged – and then adds a finite amount of symbols to the top of the stack ("push"). The symbol read from the stack serves as additional input to the agent. Formally, the agents' protocol is captured by the 4-tuple $\Pi = \langle Q, s_0^a, \Gamma, \delta \rangle$, where Q is the finite set of *states*, $s_0^a \in Q$ is the *initial state* of agent a, Γ is the finite *stack alphabet*, and $\delta : Q \times 2^Q \times \Gamma \cup \{\varepsilon\} \to 2^{Q \times \Gamma^* \times \{N,E,S,W,P\}}$ is the *transition function*. Suppose that at time $t_a(i)$, agent a is in state $q \in Q$, positioned in cell $c \in \mathbb{Z}^2$, and the top-most symbol on the stack is $\gamma \in \Gamma \cup \{\varepsilon\}$. Then, the state $q' \in Q$ of agent a at time $t_a(i+1)$, the word $\alpha \in \Gamma^*$ to be written to the stack, and the corresponding movement $\tau \in \{N, E, S, W, P\}$ are dictated based on the transition function δ by picking the tuple (q', α, τ) uniformly at random from $\delta(q, \gamma, Q_a)$, where $Q_a \subseteq Q$ is defined as in an FA-protocol.

1.3. Problem Setting

We consider two different variants of the problem, where the goal in both is to locate an adversarially hidden *treasure*, i.e., to bring at least one agent to the cell in which the treasure is positioned while the distance of the treasure from the origin is denoted by D. In async-ANTS, the problem is to find the treasure in an arbitrary asynchronous environment while in the sync-ANTS problem the agents operate in a synchronous environment. A FA/PDA-protocol $\mathcal{P}$ is *effective* if it allows the agents to locate the treasure in finite time if $\mathcal{P}$ is deterministic, or if the agents locate the treasure in expected finite time if $\mathcal{P}$ is randomized.

1.4. Preliminaries

For our deliberations we require a sequence of definitions. Let $\mathcal{A}$ be the set of agents. We denote by $E_a^{\mathcal{P}}(t)$ the cells that an agent a employing protocol $\mathcal{P}$ has visited until time t and furthermore $E^{\mathcal{P}}(t) = \bigcup_{a \in \mathcal{A}} E_a^{\mathcal{P}}(t)$. In the context of the sync-ANTS problem, we take the liberty to write $E_a^{\mathcal{P}}(i)$ for a (then global) step i as shorthand for $E_a^{\mathcal{P}}(t_a(i))$ and analogous for $E^{\mathcal{P}}(i)$. We omit $\mathcal{P}$ in the previous expressions if the considered protocol is clear from the context.

1.5. Non-Distinguishable Cells

As mentioned above, in our model the agents *cannot* distinguish between any two cells on the grid. In particular, an agent cannot distinguish the origin from any other cell. This is in clear contrast to most previous work, where the origin usually is a special cell that can be distinguished by the agents. Here we will briefly justify why we deviate from the established literature in this respect.

The goal of this paper is to determine the minimum number of agents required to locate the treasure under different model variations. Previous work mainly considered large numbers of ants where a distinguishable origin can easily be emulated by leaving a single dedicated agent at the origin without affecting the asymptotic parameters of the algorithm. Clearly, such an assumption does not effectively change the power of a model when one is interested in asymptotics.

Since we consider small, i.e., constant number of agents, the situation is slightly different. As some of the algorithms presented in the previous section employ one agent solely to mark the origin, one could argue analogously to the case of many agents and allow the agents to distinguish the origin and in return raise the minimum number of required agents by one. However, we have also presented algorithms that do not use an agent to mark the origin and therefore would not benefit from such a modification. It hence seems that a distinguishable origin is not an essential requirement for effective algorithms and thus we decided to consider the weaker model, thereby leaving it to the discretion of the algorithm designer whether or not a distinguishable origin is needed.

As we do not want the agents to infer any information on *where* on the grid they are standing simply based on information that they can observe from standing on a certain cell, we chose the infinite grid as the underlying topology. Other topologies, for example a quarter plane or a an infinite strip, do not fulfill this criterion, as agents can distinguish between cells on the border and other cells simply by looking at the degree of the cell. As mentioned above, we wanted to study the weakest model, and we note that the presented algorithms can easily be modified in such a way that certain agents are substituted by landmark information.

1.6. Outline and Results

The rest of the paper is structured as follows. First, in Section 2, we study the asynchronous and deterministic version of our model and show that 4 agents,

Problem	**FA**				**PDA**			
	sync		async		sync		async	
	det	rand	det	rand	det	rand	det	rand
One agent		$\times^{7}$		$\times^{7}$	$\times^{8}$	✓^{9}	$\times^{8}$	✓^{9}
Two agents	$\times^{4}$	?	$\times^{4}$	?	✓^{5,6}		✓^{6}	
Three agents	✓^{2}	✓^{3}	?	✓^{3}				
Four agents			✓^{1}					

Table 1: The symbol $\times$ indicates that the given combination does not allow for an effective protocol while ✓ states that there does exist an effective protocol. Empty cells follow immediately from other entries while cells marked with ? represent open problems. The numbers in the superscript refer to the theorem establishing the respective result.

controlled by finite automata, are enough to discover the treasure. In Section 3, we study the randomized and the synchronous variants of the model and show that under both of these variants, 3 agents are enough to discover the treasure. Then, in Section 4, we show that two agents controlled by a PDA are enough to discover the treasure and given access to random bits, we show in Section 5 that the treasure can be discovered by a single PDA controlled agent.

Then, we contrast these results by showing in Section 4 that 2 deterministic finite automata are not enough to discover the treasure. Finally, in Section 5, we show that neither one randomized finite automaton nor a single deterministic PDA is enough to find the treasure. Our results are summarized in Table 1.

2. Four Agents

The goal of this section is to solve the async-ANTS problem without using randomization. We provide a simple protocol for four FA-agents that uses three of the four agents as landmarks for the fourth agent. The fourth agent discovers the whole grid in a spiraling fashion with increasing distance to the origin.

We begin by giving an informal description of the protocol. The landmark agents, referred to as Guides, position themselves in a triangle around the origin and after getting a signal from the searching agent, called the Explorer, move step by step further away from the origin. The Explorer moves to the Guides one by one signaling them to expand the triangle. This way the Explorer is able to guarantee that it can always reach one Guide after meeting another by simply walking a (possibly diagonal) straight line, even after the Guides are within a super-constant distance from each other and the origin.

All three Guides have specific roles and therefore we give them task-specific names: NorthGuide, WestGuide and EastGuide. The agents execute the following protocol, which is illustrated in Figure 1. The protocol is initialized by the NorthGuide moving once north, the WestGuide moving once west and the EastGuide moving once east. After the Explorer notices that the origin is empty, it moves once north.

NorthGuide. When the NorthGuide meets a WaitingExplorer, it moves once north.

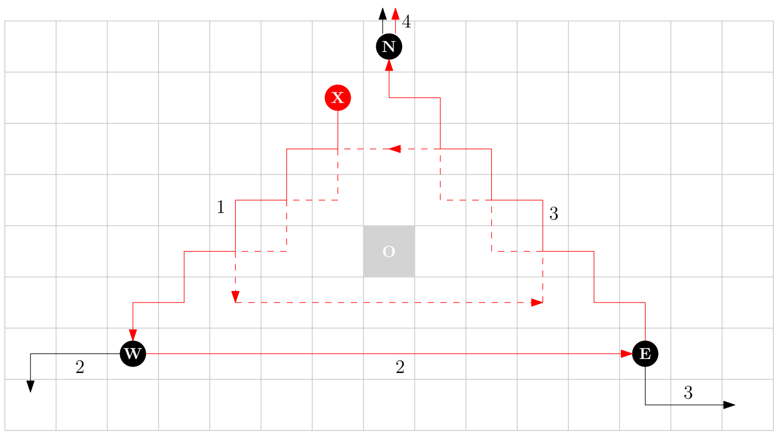

Figure 1: Four agents are discovering the grid and currently are performing a triangle search in distance 3. The origin is denoted by a gray square, the Explorer (X) by a red circle and the NorthGuide (N), WestGuide (W) and EastGuide (E) by black circles labeled with the corresponding initial letters. The numbers indicate the order of movements, i.e., moves along the arrow labeled with i are performed only after the moves along the arrow labeled with $i-1$ are finished. The dashed red line indicates the path of the Explorer in distance 2.

WestGuide. When the WestGuide meets a WaitingExplorer it moves once west and becomes a MovingWestGuide. The MovingWestGuide first moves once west and then once south and becomes a WestGuide again.

EastGuide. When the EastGuide meets a WaitingExplorer it moves once south and becomes a MovingEastGuide. The MovingEastGuide moves twice east and becomes again an EastGuide.

Explorer. The Explorer continuously performs *triangle searches* in increasing distances. It continuously moves into a given direction, starting with south-west (by alternatingly moving south and west). When the Explorer meets a WestGuide, it changes its moving direction to east and becomes a WaitingExplorer. When it meets an EastGuide, it changes the direction to north-west and becomes a WaitingExplorer. Finally, when the Explorer meets a NorthGuide, it changes its moving direction to south-west (alternates between west and south) and becomes a WaitingExplorer. Notice that the Explorer meets the NorthGuide in the starting position of the triangle search in the next distance. Whenever the Explorer meets a MovingWestGuide or a MovingEastGuide in cell c, it waits until c is empty before continuing to move.

WaitingExplorer. When the WaitingExplorer resides in a cell that does not contain an EastGuide, a NorthGuide, or a WestGuide, it becomes an Explorer and continues moving.

We index the triangle searches by their distances, i.e., if the Explorer meets the NorthGuide in cell $(0, i)$ and starts moving south-west, we index the corresponding triangle search by index i and denote it by TS_i. A triangle search in distance i starts when the Explorer leaves cell $(0, i)$ by moving west and ends when the Explorer meets a NorthGuide. Furthermore, we say that TS_i *works correctly*, if the Explorer meets the WestGuide only in cell $(-2i+1, -i+1)$, the EastGuide only in cell $(2i-1, -i+1)$ and the NorthGuide only in cell $(0, i+1)$ during TS_i.

Lemma 1. *Every triangle search works correctly.*

Proof. Consider TS_1. Initially, all the Guides are located in cells adjacent to the origin. By the design of our protocol, the Explorer first makes sure that the NorthGuide goes into cell $(0, 2)$. After this, it moves south-west and reaches the WestGuide in cell $(-1, 0)$. Then it travels east and reaches the EastGuide in cell $(1, 0)$. From there, it travels north-west and meets the NorthGuide in cell $(0, 2)$. Thus, the claim holds for TS_1.

Assume then that the claim holds for TS_{i-1} and consider TS_i. The Explorer starts moving south-west from cell $(0, i)$. According to the induction assumption, the WestGuide is located in either $(-2i+2, -i+2)$, $(-2i+1, -i+2)$ or $(-2i+1, -i+1)$. Since the Explorer moves diagonally, it has to pass all of these cells. According to the design of our algorithm, it does not overtake the MovingWestGuide, i.e., the MovingWestGuide reaches its destination before the Explorer, and therefore the Explorer meets the WestGuide in cell $(-2i+1, -i+1)$.

Similarly, when the Explorer starts moving towards east, the correctness of the previous triangle ensures that the MovingEastGuide reaches the cell $(2i-1, -i+1)$ before the Explorer. After meeting the EastGuide, the Explorer starts moving diagonally towards the starting point and reaches it after $2i$ movements. Since the Explorer moves north in the next step, it meets the NorthGuide in cell $(0, i+1)$. ☐

To show that the treasure eventually gets discovered, we need two more auxiliary observations. First, we show that every cell in distance d is discovered latest during TS_{d+1}. Second, we show that each triangle search finishes within finite time. We call the set of cells along which the Explorer moves during TS_i the path of rectangle search i.

Observation 2. *Every cell c within distance d to the origin is discovered latest during TS_{d+1}.*

Proof. We prove the claim by induction on the distances of the cells, i.e., we show that all cells within distance d are contained in a triangle search with index at most $d+1$. The base case is clear since the origin is contained within the path that the Explorer moves during TS_1.

Assume then that the claim holds for all cells in distance d. By the design of the triangle search protocol, the path of TS_{i+1} contains all the cells adjacent to the cells in the path of TS_i that are not discovered during TS_i. See Figure 1 for illustration. Therefore, all cells in distance $d+1$ are discovered latest during TS_{d+2}. ☐

Observation 3. *Every triangle search ends within finite time.*

Proof. Let t be the time when TS_i starts for some $i > 0$. By Lemma 1, we know that TS_{i-1} worked correctly and therefore we know that the WestGuide reaches cell $(-2i+1, -i+1)$ and the EastGuide reaches cell $(2i-1, -i+1)$ latest by time $t+3$. Therefore, latest by time $t+3+4i$, the Explorer meets the WestGuide in cell $(-2i+1, -i+1)$. By time $t+3+4i+2$, the WestGuide has left the cell and the Explorer can continue moving east. By time $t+5+4i+4i+2$, the Explorer turns towards the NorthGuide and finally reaches its cell by time $t+7+8i+4i$ ending the triangle search. □

We can now combine the results from this section. Recall that D is the distance to the treasure. By Observation 2, the treasure is found latest during TS_{D+1}. As the duration of each search is finite by Observation 3 and by Lemma 1 each triangle is eventually searched, we get the following theorem.

Theorem 1. *There exists an effective deterministic FA-protocol for async-ANTS for $n = 4$.*

3. Three Agents

3.1. Deterministic Protocol for sync-ANTS

In this section, we first show that we can get rid of one of the FA-agents by giving the agents a common notion of time. In other words, if we assume that the execution of the algorithm is synchronous, three agents suffice to discover the treasure. Our goal is to prove the following theorem.

Theorem 2. *There exists an effective deterministic FA-protocol for sync-ANTS for $n = 3$.*

The idea of the three-agent protocol is similar to the protocol from Section 2. Again, one of the agents, the Explorer, performs the actual searching and the two other agents work as Guides. The task of one of the Guides, called OriginGuide, is simply to stand still and mark the origin throughout the execution. The task of the other Guide is to tell the Explorer when it hits an axis. On the first round of the execution, the Explorer and the other Guide move one step north to cell $(0, 1)$ and then start the execution of the following protocol.

Explorer. The Explorer repeatedly performs *rectangle searches* in increasing distances. It starts the first rectangle search in distance 1 by diagonally moving south-west, i.e., alternating between moving west and south. When it meets a Guide, it alters its movement direction by 90° counter-clockwise. At the end of a complete rectangle (i.e., when meeting a Guide again at the starting point), it moves one step outwards starting a new rectangle search with a larger distance. During a rectangle search in distance d, the Explorer discovers all cells that have distance d to the origin.

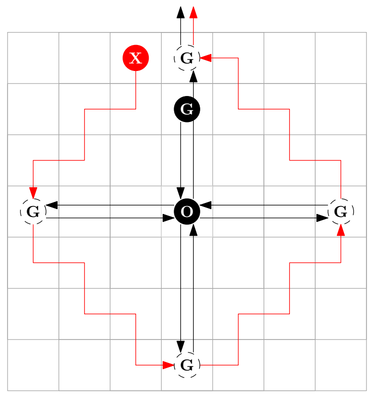

Figure 2: Three agents can discover the entire grid under a synchronous environment. The dashed circles indicate the locations where the Explorer (X) meets the Guide (G). The OriginGuide (O) marks the origin.

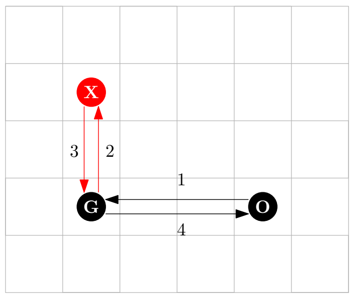

Figure 3: Three agents are performing a geometric search on the north-west quarter plane. Moves along the black arrows are executed by both the Explorer (X) and the Guide (G) while the OriginGuide (O) states at the origin. Moves along the red arrows are executed only by the Explorer.

Guide. The Guide starts by moving towards the OriginGuide that marks the origin. When it meets the OriginGuide, it alters its direction by 90° clockwise and moves outwards. When it meets the Explorer, it turns around and moves inwards towards the OriginGuide. The Guide also moves one step north with the Explorer when they meet in the end of searching a rectangle and starts walking towards the OriginGuide afterwards.

The execution of our protocol is illustrated in Figure 2. To prove Theorem 2, we only need to show that every time the Explorer enters a cell on an axis, it meets a Guide. To see why this is sufficient, consider any cell c on the plane with distance d to the origin. Then c is searched (latest) during rectangle search in distance d. Therefore, assuming that each rectangle search is performed correctly, the whole plane is eventually discovered.

It is fairly easy to see that the Explorer and the Guide never fail to meet. Consider round r when the Explorer and a Guide meet on an axis during rectangle search in distance d. Then the distance that both of them have to move until the next meeting point is $2d$. Since both agents move exactly once per round, the claim follows. Note that the assumption of a synchronous environment is crucial here.

3.2. Randomized Protocol for async-ANTS

We now show that if we are not restricted to deterministic state machines but allow randomization, we can find the treasure under an asynchronous environment with only 3 FA-agents. The fundamental idea behind our randomized protocol is that the agents use a fair coin to determine which cells to discover.

Again, we have two Guides and one Explorer and the task of one of the agents, the OriginGuide, is to simply stay in the origin. The Explorer performs the actual searching and starts by uniformly at random choosing either (north, east), (east, south), (south, west) or (west, north), i.e., it randomly chooses a quarter plane. Then, the Explorer performs a *geometric search* on that quarter plane.

Consider the case of choosing (east, south) as the quarter-plane (the search in the other quarter-planes works analogously). The Guide and the Explorer execute the following protocols.

Explorer. The Explorer starts by moving once east. Then on every step the Explorer tosses a fair coin and if it shows heads, it moves east. When the coin shows tails, the Explorer stops and becomes a WaitingExplorer until its cell is occupied by a WaitingGuide. When the WaitingGuide appears, the WaitingExplorer moves one cell south, becomes an Explorer, and continues tossing coins but now moves one cell south every time the coin shows head instead of east. When the coin shows tails, the Explorer turns back, i.e., starts moving north. After the Explorer reaches a cell with a WaitingGuide, it stops and moves west (until it reaches an OriginGuide) whenever its cell contains no WaitingGuide.

Guide. The Guide moves east on every step if its cell is not occupied by an Explorer. When it meets a WaitingExplorer, it turns into a WaitingGuide. When the WaitingGuide meets an Explorer, it becomes a Guide again and moves west whenever its cell is not occupied by an Explorer until it meets an OriginGuide.

After all the agents reach the origin, they restart the process. The protocol is illustrated in Figure 3. It is easy to see that each geometric search has a finite duration with probability 1 since the Explorer throws a finite number of heads in every search with probability 1. Assume that the number of heads is finite. Then the Explorer becomes a WaitingExplorer in finite time. After the Explorer becomes a WaitingExplorer, the Guide moves towards the cell of the WaitingExplorer in every step and therefore reaches it in finite time. Similarly, the Explorer returns to the WaitingGuide in finite time and they both reach the OriginGuide in finite time.

Theorem 3. *There exists an effective randomized FA-protocol for async-ANTS for $n = 3$.*

Proof. Assume that the treasure is located in cell $c = (x, y)$ in the north-east quarter plane with $D = x + y$. Let us index the geometric searches, i.e., the iterations of the algorithm, by the positive integers. Clearly, the protocol is defined so that if the treasure is found in search i, then search $j > i$ is not needed, however, for the sake of the analysis, we assume that the agents keep performing the searches indefinitely and bound the time until the treasure is found – let T be the random variable that captures this time. Given this view, we know that search i is independent of all searches other than i.

Let A_i be the event that the Explorer finds the treasure in search i. This happens if it chooses the right quarter plane, throws heads exactly $x - 1$ times before throwing tails once and then throws heads $y - 1$ times. Hence, $\Pr(A_i) = \frac{1}{4} \cdot 2^{-(x-1)} \cdot \frac{1}{2} \cdot 2^{-(y-1)} = 2^{-(D+1)}$. Let $B_i = \neg A_1 \wedge \cdots \neg A_{i-1} \wedge A_i$ be the event that the treasure is found in search i and not in any search $j < i$. Let L_i be the random variable that measures the number of distinct cells the Explorer visits in search i, i.e., the length of the path along which the Explorer moves during search i. We rely on the following equations that hold for every $i \geq 1$ and $1 \leq j < i$:

(1) $\Pr(A_i) = 2^{-(D+1)}$
(2) $\Pr(B_i) = (1 - 2^{-(D+1)})^{i-1} 2^{-(D+1)}$
(3) $\mathbb{E}[L_i \mid B_i] = \mathbb{E}[L_i \mid A_i] = \mathcal{O}(D)$
(4) $\mathbb{E}[L_j \mid B_i] = \mathbb{E}[L_j \mid \neg A_j] = \mathcal{O}(1)$

Therefore,

$$\begin{aligned}
\mathbb{E}[T] &= \sum_{i=1}^{\infty} \mathbb{E}[T \mid B_i] \cdot \Pr(B_i) \\
&= \sum_{i=1}^{\infty} \Big(\sum_{j=1}^{i-1} \mathbb{E}[L_j \mid B_i] + \mathbb{E}[L_i \mid B_i] \Big) \cdot (1 - 2^{-(D+1)})^{i-1} 2^{-(D+1)} \\
&= \sum_{i=1}^{\infty} (\mathcal{O}(i) + \mathcal{O}(D)) \cdot (1 - 2^{-(D+1)})^{i-1} 2^{-(D+1)} \\
&= 2^{-(D+1)} \cdot \sum_{i=1}^{\infty} \mathcal{O}(i) \cdot (1 - 2^{-(D+1)})^{i-1} \\
&\qquad + \mathcal{O}(D) \cdot 2^{-(D+1)} \cdot \sum_{i=1}^{\infty} (1 - 2^{-(D+1)})^{i-1} \\
&= 2^{-(D+1)} \cdot \mathcal{O}(2^{2D}) + \mathcal{O}(D) \cdot 2^{-(D+1)} \cdot 2^{D+1} = \mathcal{O}(2^D) \ .
\end{aligned}$$

□

4. Two Agents

Our goals in this section are to show, on the negative side, that two deterministic FA-agents *cannot* solve sync-ANTS, and, on the positive side, that one deterministic FA-agent together with one deterministic PDA-agent *can* solve sync-ANTS.

4.1. No Deterministic FA-Protocol

We start off with proving the first result. Before doing so, we define the notion of a *band* in $\mathbb{Z}^2$. A band is the discrete version of a fat line in Euclidean space, i.e., the set of cells that have at most a certain distance from a line.

Definition. A *band* $B = (s, m, e)$, $s = (s_x, s_y) \in \mathbb{Z}^2$ with slope $m = (m_x, m_y) \in \mathbb{Z}^2$ of extent $e \in \mathbb{N}_{>0}$ consists of all cells c for which there exists a point $p = (s_x + \lambda m_x, s_y + \lambda m_y)$ for some $\lambda \in \mathbb{R}$ such that $\|c - p\|_1 \leq e$ where $\|x\|_1$ denotes the ℓ_1-norm of x.

Observation 4. *Let $\mathcal{B}$ be a finite set of bands with finite extent. Then $\mathbb{Z}^2 \setminus \bigcup_{B \in \mathcal{B}} B \neq \emptyset$.*

Proof. Assume for the sake of a contradiction that the bands in $\mathcal{B}$ cover $\mathbb{Z}^2$ completely. Let e^* be the maximum extent of the bands in $\mathcal{B}$. Consider a square region S of $\mathbb{Z}^2$ with ℓ^2 cells for $\ell > 2|\mathcal{B}|e^*$ and a fixed band $B = (s, m, e) \in \mathcal{B}$. Assume wlog. that $|m_x| \leq |m_y|$. Observe that $|B \cap S| \leq \ell \cdot 2e^*$ since S vertically extends over ℓ cells and the horizontal width of $B \cap S$ is at most $2e^*$. Let $A = \bigcup_{B \in \mathcal{B}} B$ and we get $|A \cap S| \leq 2|\mathcal{B}|e^* \cdot \ell < \ell^2 = |S|$. Thus, the bands in $\mathcal{B}$ do not even cover the cells in S, a contradiction. $\square$

We denote by $M(\mathcal{P}) = (t_i)_{i>0}$ the strictly increasing sequence of all points in time when two agents meet during the execution of protocol $\mathcal{P}$. Furthermore, we denote the coordinates of agent a at time t by $C_a(t)$. An important ingredient for the proof is the following lemma, which holds for an arbitrary amount of agents.

Lemma 5. *If $\mathcal{P}$ is an effective deterministic FA-protocol for* sync-ANTS*, then $|M(\mathcal{P})| = \infty$.*

Proof. Assume for the sake of contradiction that $\mathcal{P}$ is an effective deterministic protocol with finite $|M(\mathcal{P})|$. Thus, there exists a largest point in time $t^* = \max(M(\mathcal{P}))$ when two agents meet and after which no two agents meet anymore and the number of cells explored until t^* is finite. Consider now agent a and let q be the state that has been entered by agent a twice after t^* at the earliest time. Let $(t_i)_{i>0}$ be the strictly increasing sequence of points in time after t^* when a enters state q and denote $I_i = [t_i, t_{i+1}]$. Observe that the behavior of a in each interval I_i is identical, hence a will keep on repeating the same transitions and movements as in I_1 forever. Observe further that a can only move a finite distance in each I_i as it has a finite length.

Consider the vector $v_i(a) = C_a(t_{i+1}) - C_a(t_i)$ describing the net-translation of a during I_i and observe that by the above argument $v_i(a) = v_1(a)$ for all $i > 0$. There are two cases: If $v_1(a) = 0$, then agent a explores only a constant amount of cells for $t \to \infty$. If $v_1(a) \neq 0$, then a exhibits a net-movement into the direction of $v_1(a)$ in each I_i and since it only explores a constant amount of cells in each I_i, agent a explores only cells in a band with finite width after t^*. By Observation 4, the agents cannot explore all cells in $\mathbb{Z}^2$ and the claim follows. $\square$

Theorem 4. *There exists no effective deterministic FA-protocol for* sync-ANTS *for $n = 2$.*

Proof. Assume for the sake of contradiction that $\mathcal{P}$ is an effective deterministic protocol for two agents a_1 and a_2. By Lemma 5 we know that $|M(\mathcal{P})| = \infty$. Let Q_1 and Q_2 be the set of states of the two FAs controlling a_1 and a_2. We denote by $Q_1(t) \in Q_1$ and $Q_2(t) \in Q_2$ the state of agent a_1 and a_2 at time t and further $Q(t) = (Q_1(t), Q_2(t))$. Observe that since $|M(\mathcal{P})| = \infty$, there must be a pair of states $(q_1, q_2) \in Q_1 \times Q_2$ such that the sub-sequence $T = (\tau_i)_{i>0}$ of $M(\mathcal{P})$ that consists of all $\tau \in M(\mathcal{P})$ such that $Q(\tau) = (q_1, q_2)$, is infinite. We denote the intervals $I_i = [\tau_i, \tau_{i+1}]$ and observe that a_1 and a_2 (individually) perform exactly the same state transitions and movements in each interval I_i

(agent a_1 and a_2 might meet between τ_i and τ_{i+1} in different states, but their behavior is fully determined by their states at time τ_i). Thus, there is a fixed vector $v = C_{a_1}(\tau_{i+1}) - C_{a_1}(\tau_i)$ representing the translation of the meeting cell of a_1 and a_2 during some I_i and furthermore a fixed constant $\vartheta > 0$ such that $\tau_{i+1} - \tau_i = \vartheta$. Consequently, a_1 and a_2 can only explore cells in a band with finite width after τ_1. Since $E(\tau_1)$ is finite, Observation 4 yields a contradiction. □

4.2. Deterministic FA/PDA-Protocol for sync-ANTS

The second result of this section establishes that while two agents controlled by a FA do not allow for an effective deterministic protocol for sync-ANTS, one FA-agent and one PDA-agent do so.

The protocol is essentially an adapted version of the protocol from Section 3.1. The Explorer behaves identically to Section 3.1 and performs rectangle searches with increasing distances to the origin. The second PDA-agent replaces the two Guides by walking along the axis in order to signal to the Explorer when the search in a quarter-plane is complete and it should therefore alter its movement direction. The trick here is that the Guide tracks its distance from the origin using the stack. More precisely, the Guide pushes a symbol onto the stack whenever it performs a movement outwards on one of the axes and pops one symbol from the stack whenever it moves towards the origin. Using this trick, the Guide can detect when it has arrived at the origin by verifying whether the stack is empty, i.e., the read symbol is ε. Then the algorithm works as follows:

At time $t = 0$, the Guide and the Explorer both move one cell north (and the Guide records this move on the stack). Whenever the two agents are located together on the north-axis in cell $(0, d)$, the Explorer starts a diagonal walk towards south-west while the Guide moves south towards the origin until it arrives there, which it can track using the stack. Upon arriving there, it moves west until it meets the Explorer. As the length of the two (different) paths from cell $(0, d)$ to cell $(-d, 0)$ is equal, both the Guide and the Explorer arrive in cell $(0, -d)$ at the same time. Now the Explorer changes its movement direction and the Guide moves back to the origin after which it moves south to meet the Explorer on the south-axis in cell $(0, -d)$. They repeat this process to meet on the west-axis in cell $(d, 0)$ and on the north-axis in cell $(0, d)$. When the Explorer has completed the rectangle search of level d by arriving at cell $(0, d)$ again, it moves together with the Guide to cell $(0, d + 1)$ and the search of level $d + 1$ begins.

It is easy to see that the above algorithm guarantees that the Explorer meets the Guide every time it crosses an axis and that therefore any level d is explored in finite time.

Theorem 5. *There exists an effective deterministic protocol for sync-ANTS for $n = 2$ that uses one FA-protocol and one PDA-protocol.*

4.3. Deterministic PDA-Protocol for async-ANTS

Since two PDAs can simulate a Turing machine [20] by using both their stacks to represent the infinite band of the Turing machine, it is not too surprising that two PDAs allow for an effective deterministic protocol for async-ANTS.

The two agents a and b employ the following protocol: Both agents walk "hand-in-hand", i.e., have a distance of at most 1 at all times, and perform a spiral search with increasing distances from the origin (cf. Section 3.1). At any time during the execution, they maintain the invariant that the sum of the number of symbols on both stacks equals their distance from the origin. They start from the cell $(0, 1)$ with the stack of agent a containing one symbol. When the two agents start a spiral search from cell $(0, i)$, agent a has i symbols on is stack. When a and b walk south-west, agent a removes a symbol from its stack every other step while agent b pushes one symbol to its stack every other step. When the stack of agent a is empty, agent b's stack contains i symbols and the agents have arrived at the cell $(-i, 0)$ on the west-axis. Then they reverse their roles and move together to the south, east, and again north-axis in the same fashion to finish the search in distance i. Thereafter, they move one cell north, push one additional symbol to the stack to account for the increased distance and start a new search in distance $i + 1$. It is easy to see that this protocol can be implemented to work in an asynchronous environment and guarantees that the two agents locate the treasure.

Theorem 6. *There exists an effective deterministic PDA-protocol for async-ANTS for $n = 2$.*

5. One Agent

In this section we show that neither a single randomized FA-agent nor a single deterministic PDA-agent can find the treasure in finite time while a randomized PDA-agent is able to do so.

5.1. No Randomized FA-Protocol

Consider a single agent who is controlled by a finite state machine. The movements the agent performs on the grid can be described by a Markov chain, where we simply assign the state set of the Markov chain to be the states of the finite state machine and the transition probabilities to be the probabilities assigned to the corresponding state transitions. Clearly, this Markov chain is finite and all state transition probabilities are constants. Therefore, it must contain an irreducible subset H of states that are entered within constant amount of state transitions with a non-zero constant probability; for the rest of the section, we condition on this event and focus on the restriction of the Markov chain to the states of H. If we show that the expected time to reach the treasure is infinite under this event, it follows that the expected time to reach the treasure is infinite in general, since this event occurs with constant probability.

Let s be the first state in H visited by the Markov chain. Let d be the distance of the agent from the origin when the Markov chain visits state s for the first time and recall that d is bounded from above by some constant. Since H is an irreducible set of states with finite cardinality $|H|$, standard Markov chain theory (see, e.g., [21]) ensures that state s is visited infinitely often with probability 1 and the expected time between any two such visits is finite.

For our purposes, it is enough to study the movements in one dimension and therefore, we project the location of the agent on the x-axis and ignore its movements in the y-dimension. So, in what follows we focus on a single agent that traverses $\mathbb{Z}$ and assume towards contradiction that the agent hits every point $x \in \mathbb{Z}$ in finite expected time.

Let $\mathbb{Z}_s$ be the set of integers $x \in \mathbb{Z}$ such that the agent has a positive probability to visit x while in state s. Since H is finite, there exist a positive integer m and a non-negative integer k such that $\mathbb{Z}_s = \{zm + k \mid z \in \mathbb{Z}\}$. Let $X_i \in \mathbb{Z}$, $i = 1, 2, \ldots$, be the (x-coordinate of the) location of the agent at the ith time the Markov chain visits state s and let $Z_i = (X_i - k)/m$.

The key observation now is that the irreducibility of the Markov chain controlling the movements of the agent implies that the agent hits every point $x \in \mathbb{Z}_s$ *while in state* s in finite expected time. In other words, the random variable $T(z) = \min\{i \geq 1 \mid Z_i = z\}$ has finite expectation for every $z \in \mathbb{Z}$.

The stochastic process $\{Z_i\}_{i \geq 1}$ is, by itself, Markovian, i.e.,

$$\Pr(Z_n = z_n \mid Z_1 = z_1, \ldots, Z_{n-1} = z_{n-1}) = \Pr(Z_n = z_n \mid Z_{n-1} = z_{n-1}) \,.$$

This process belongs to a family of stochastic processes that have been characterized by Feller [22, p. 396, Theorem 2] based on the random variable $Y = Z_{i+1} - Z_i$, concluding that:

(1) if $\mathbb{E}[Y] > 0$, then Z_i drifts to the right and $\mathbb{E}[T(z)] = \infty$ for any $z < -d$;
(2) if $\mathbb{E}[Y] < 0$, then Z_i drifts to the left and $\mathbb{E}[T(z)] = \infty$ for any $z > d$; and
(3) if $\mathbb{E}[Y] = 0$, then Z_i is null-recurrent and $\mathbb{E}[T(z)] = \infty$ for any z such that $|z| > d$.

In all three cases we reach a contradiction to the assumption that $\mathbb{E}[T(z)]$ is finite for all $z \in \mathbb{Z}$, thus establishing the following theorem.

Theorem 7. *There exists no effective randomized FA-protocol for* sync-ANTS *for $n = 1$.*

5.2. No Deterministic PDA-Protocol

Consider a single agent controlled by a *deterministic* PDA-protocol. We denote by $S(i)$ the size of the stack, i.e., the number of symbols on the stack (directly) after step i and by $C(i) = (q, \gamma)$ the tuple of the state $q \in Q$ and the top-most stack symbol $\gamma \in \Gamma$ (directly) after step i. Let $\mathcal{C} = Q \times \Gamma$ be the set of all configurations and observe that $|\mathcal{C}|$ is constant. As the behavior of a PDA is fully determined by its state and the top-most stack symbol, the following observation is immediate.

Observation 6. *Let $0 < i_1 < i_2$ be two different steps with $C(i_1) = C(i_2)$ and let i_2 be the smallest such index. If $S(i) \geq S(i_1)$ for all $i_1 \leq i \leq i_2$, then $C(j) = C(j + k \cdot (i_2 - i_1))$ for all $i_1 \leq j \leq i_2$ and $k \in \mathbb{N}_0$.*

Note that the observation also implies that the agent executes the identical sequence of actions between step i_1 and i_2.

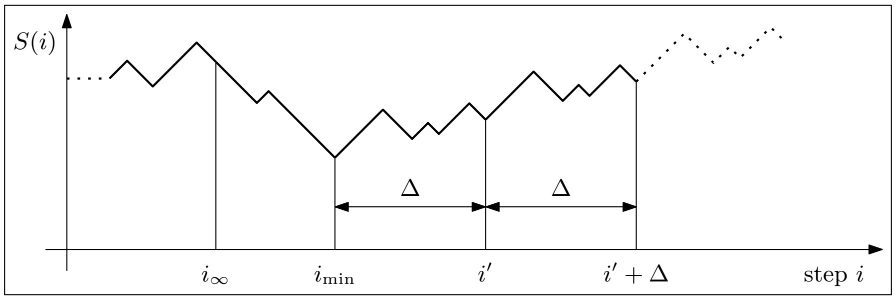

Figure 4: The size $S(i)$ of the stack varies for the different steps. All configurations entered after step i_∞ are entered infinitely often. The stack exhibits its minimal size after i_∞ at step $i_{\min}$ while $C(i_{\min})$ is entered again for the first time at time i'. Then the PDA will keep repeating its behavior after $i_{\min}$ with period $\Delta = i' - i_{\min}$.

Observe that, since any protocol must be able to run for an arbitrary time, we can partition the set $\mathcal{C}$ into the configurations $\mathcal{C}_f$ containing all configurations that are entered finitely often and the configurations $\mathcal{C}_\infty$ that are entered infinitely often during the execution of a given protocol. Observe that there exists step i_∞ such that $C(i) \in \mathcal{C}_\infty$ for any step $i > i_\infty$. The following lemma essentially states that after a certain step $i_r > i_\infty$, the PDA will keep on repeating its behavior with a finite period Δ (see Figure 4 for an illustration).

Lemma 7. *There exists an index $i_r > i_\infty$ and a period $\Delta \in \mathbb{N}_0$ such that for all steps i with $i_r \leq i < i_r + \Delta$ we have $C(i + k \cdot \Delta) = C(i)$ for all $k \in \mathbb{N}_0$.*

Proof. Let $s_{\min} \in \mathbb{N}_0$ be the minimum stack size after i_∞ and let $i_{\min}$ be the smallest index $i > i_\infty$ for which $S(i) = s_{\min}$. Let $i' > i_{\min}$ be the smallest step such that $C(i') = C(i_{\min})$. By definition of $i_{\min}$ there exists no index $i > i_{\min}$ with $S(i) < S(i_{\min})$. Thus, $i_{\min}$ and i' satisfy the preconditions of Observation 6 and the claim follows for $i_r = i_{\min}$ and $\Delta = i' - i_{\min}$. $\square$

As the PDA keeps on repeating its behavior after step i_r with constant period Δ, the agent can only explore cells in a band of finite width after i_r. As i_r is finite and thus $E(i_r)$ is also finite, Observation 4 implies the following theorem.

Theorem 8. *There exists no effective deterministic PDA-protocol for sync-ANTS for $n = 1$.*

5.3. Randomized PDA-Protocol for async-ANTS

The randomized protocol is an adapted version of the randomized FA-protocol for three agents from Section 3.2. There, one agent repeatedly performs geometric searches to a random cell in a geometrically distributed distance. It uses the two other agents to find its way back to the origin in order to start the next iteration of the search. A single agent employing a randomized PDA-protocol can do the same by using the stack to record its distance to the origin and thereby, it can perform a geometric search and then return to the origin for the next iteration. More precisely, the agent performs a geometric search as in Section 3.2 but whenever moving north/east/south/west, it pushes N/E/S/W, respectively, to the stack. When one geometric search ends, the agent can retrace its steps by walking north/east/south/west when reading S/W/N/E, respectively, and ends up at the origin when the stack is empty. Then, it can start the next iteration. To show that time required to find the treasure with this protocol is finite, we can simply observe that the PDA-protocol is never slower than the protocol with 3 FA-agents and therefore, we get the following theorem directly from Theorem 3.

Theorem 9. *There exists an effective randomized PDA-protocol for async-ANTS for $n = 1$.*

6. Returning to the Origin

In this section we briefly explain the techniques through which the previously mentioned protocols can be amended in order to guarantee that, upon locating the treasure, all agents return to the origin in a timely manner, i.e., with a constant multiplicative overhead in terms of the runtime.

For all deterministic protocols, the idea is simply that upon locating the treasure, the agent(s) invert the search and progressively move closer towards the origin in the same manner as they moved further away from the origin in the search stage. More concretely, when the Explorer locates the treasure, it first finishes the search of the current distance from the origin (the current triangle or diamond in the 4-ant and 3-ant protocol, respectively) to move all guides into a well-defined position. Then it starts a walk along the triangle/diamond in the next distance but notifies the Guides that it meets on the way that the treasure has been found and that they should also retrace their steps back towards the origin. Clearly, this technique ensures that all the agents eventually end up at the origin. In the two deterministic protocols involving PDAs, returning to the origin is even simpler. After locating the treasure, the agents continue until they arrive at the next axis and then, knowing the distance to the origin, they simply walk back along the axis until they arrive at the origin.

As both randomized protocols involve that all agents repeatedly return to the origin, the agents can just follow the same procedure upon locating the treasure.

7. Conclusion

The variety of results of this paper are summarized in Table 1. While our findings almost completely cover the landscape of problem configurations, Table 1 essentially shows two gaps, which, in our opinion, represent interesting open problems: Can two agents controlled by a randomized FA solve the synchronous or asynchronous version of the ANTS problem? Is there an effective FA-protocol for async-ANTS for three agents when no random bits are available?

We conjecture that the answer two both questions is “no”, but our efforts to prove this have not been fruitful, yet.

Bibliography

[1] Y. Emek, T. Langner, J. Uitto, R. Wattenhofer, Solving the ANTS Problem with Asynchronous Finite State Machines, in: Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP), 471–482, 2014.

[2] O. Feinerman, A. Korman, Z. Lotker, J.-S. Sereni, Collaborative Search on the Plane Without Communication, in: Proceedings of the 31st ACM Symposium on Principles of Distributed Computing (PODC), 77–86, 2012.

[3] O. Feinerman, A. Korman, Memory Lower Bounds for Randomized Collaborative Search and Implications for Biology, in: Proceedings of the 26th International Conference on Distributed Computing (DISC), 61–75, 2012.

[4] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, D. Peleg, Graph Exploration by a Finite Automaton, Theoretical Computer Science 345 (2-3) (2005) 331–344.

[5] C. Lenzen, N. Lynch, C. Newport, T. Radeva, Trade-offs between Selection Complexity and Performance when Searching the Plane without Communication, in: Proceedings of the 33rd Symposium on Principles of Distributed Computing (PODC), 252–261, 2014.

[6] T. Langner, D. Stolz, J. Uitto, R. Wattenhofer, Fault-Tolerant ANTS, in: Proceedings of the 28th International Symposium on Distributed Computing (DISC), 31–45, 2014.

[7] I. Wagner, A. Bruckstein, From Ants to A(ge)nts: A Special Issue on Ant-Robotics, Annals of Mathematics and Artificial Intelligence 31 (2001) 1–4.

[8] S. Albers, M. Henzinger, Exploring Unknown Environments, SIAM Journal on Computing 29 (2000) 1164–1188.

[9] X. Deng, C. Papadimitriou, Exploring an Unknown Graph, Journal of Graph Theory 32 (1999) 265–297.

[10] K. Diks, P. Fraigniaud, E. Kranakis, A. Pelc, Tree Exploration with Little Memory, Journal of Algorithms 51 (2004) 38–63.

[11] P. Panaite, A. Pelc, Exploring Unknown Undirected Graphs, in: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 316–322, 1998.

[12] O. Reingold, Undirected Connectivity in Log-Space, Journal of the ACM (JACM) 55 (2008) 17:1–17:24.

[13] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovasz, C. Rackoff, Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems, in: Proceedings of the IEEE 20th Annual Symposium on Foundations of Computer Science (FOCS), 218–223, 1979.

[14] M. Aigner, M. Fromme, A Game of Cops and Robbers, Discrete Applied Mathematics 8 (1984) 1–12.

[15] R. A. Baeza-Yates, J. C. Culberson, G. J. E. Rawlins, Searching in the Plane, Information and Computation 106 (1993) 234–252.

[16] A. López-Ortiz, G. Sweet, Parallel Searching on a Lattice, in: Proceedings of the 13th Canadian Conference on Computational Geometry (CCCG), 125–128, 2001.

[17] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, R. Peralta, Computation in Networks of Passively Mobile Finite-State Sensors, Distributed Computing 18 (2006) 235–253.

[18] J. Aspnes, E. Ruppert, An Introduction to Population Protocols, in: B. Garbinato, H. Miranda, L. Rodrigues (Eds.), Middleware for Network Eccentric and Mobile Applications, Springer-Verlag, 97–120, 2009.

[19] Y. Emek, R. Wattenhofer, Stone Age Distributed Computing, in: Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC), 137–146, 2013.

[20] J. E. Hopcroft, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.

[21] D. Aldous, J. A. Fill, Reversible Markov Chains and Random Walks on Graphs, unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/∼aldous/RWG/book.html`, 2002.

[22] W. Feller, An Introduction to Probability Theory and its Applications, vol. II, Wiley, 1971.