Solving the ANTS Problem with Asynchronous Finite State Machines

Yuval Emek[1], Tobias Langner[2], Jara Uitto[2], and Roger Wattenhofer[2]

[1] Technion, Israel
[2] ETH Zürich, Switzerland

Abstract. Consider the *Ants Nearby Treasure Search (ANTS)* problem introduced by Feinerman, Korman, Lotker, and Sereni (PODC 2012), where n mobile agents, initially placed in a single cell of an infinite grid, collaboratively search for an adversarially hidden treasure. In this paper, the model of Feinerman et al. is adapted such that each agent is controlled by an asynchronous (randomized) finite state machine: they possess a constant-size memory and can locally communicate with each other through constant-size messages. Despite the restriction to constant-size memory, we show that their collaborative performance remains the same by presenting a distributed algorithm that matches a lower bound established by Feinerman et al. on the run-time of any ANTS algorithm.

1 Introduction

"They operate without any central control. Their collective behavior arises from local interactions." The last quote is arguably the mantra of distributed computing, however, in this case, "they" are not nodes in a distributed system; rather, this quote is taken from a biology paper that studies social insect colonies [16]. Understanding the behavior of insect colonies from a distributed computing perspective will hopefully prove to be a big step for both disciplines.

In this paper, we study the process of food finding and gathering by ant colonies from a distributed computing point of view. Inspired by the model of Feinerman et al. [11], we consider a colony of n ants whose nest is located at the origin of an infinite grid that collaboratively search for an adversarially hidden food source. An ant can move between neighboring grid cells and can communicate with the ants that share the same grid cell. However, the ant's navigation and communication capabilities are very limited since its actions are controlled by a randomized *finite state machine* (FSM) operating in an asynchronous environment — refer to the model section for a formal definition. Nevertheless, we design a distributed algorithm ensuring that the ants locate the food source within $\mathcal{O}(D + D^2/n)$ time units w.h.p., where D denotes the distance between the food source and the nest.[3] It is not difficult to show that a matching lower

[3] We say that an event occurs *with high probability*, abbreviated by w.h.p., if the event occurs with probability at least $1 - n^{-c}$, where c is an arbitrarily large constant.

bound holds even under the assumptions that the ants have unbounded memory (i.e., are controlled by a Turing machine) and know the parameter n.

Related Work. Feinerman et al. [10, 11] introduce the aforementioned problem called *ants nearby treasure search (ANTS)* and study it, assuming that the ants (a.k.a. *agents*) are controlled by a Turing machine (with or without space bounds) and do not communicate with each other at all. They show that if the n agents know a constant approximation of n, then they can find the food source (a.k.a. *treasure*) in time $\mathcal{O}(D + D^2/n)$. Moreover, Feinerman et al. observe a matching lower bound and prove that this lower bound cannot be matched without some knowledge of n. In contrast to the model studied in [10, 11], the agents in our model can communicate anywhere on the grid as long as they share the same grid cell. However, due to their weak control unit (a FSM), their communication capabilities are very limited even when they do share the same grid cell. Notice that the stronger computational model assumed by Feinerman et al. enables an individual agent to perform tasks way beyond the capabilities of a (single) agent in our setting, e.g., list the grid cells it has already visited or perform spiral searches (that play a major role in their upper bound).

Distributed computing by finite state machines has been studied in several different contexts including *population protocols* [4, 5] and the recent work [9] from which we borrowed the agents communication model. In that regard, the line of work closest to our paper is probably the one studying graph exploration by FSM controlled agents, see, e.g., [12].

Graph exploration in general is a fundamental problem in computer science. In the typical case, the goal is for a single agent to visit all nodes in a given graph [1, 7, 8, 15, 17]. It is well-known that random walks allow a single agent to visit all nodes of a finite undirected graph in polynomial time [2]. Notice that in an infinite grid, the expected time it takes for a random walk to reach any designated cell is infinite.

Finding treasures in unknown locations has been previously studied, for example, in the context of the classic *cow-path* problem. In the typical setup, the goal is to locate a treasure on a line as quickly as possible and the performance is measured as a function of the distance to the treasure. It has been shown that there is a deterministic algorithm with a competitive ratio 9 and that a spiral search algorithm is close to optimal in the 2-dimensional case [6]. The study of the cow-path problem was extended to the case of multiple agents by López-Ortiz and Sweet [14]. In their study, the agents are assumed to have unique identifiers, whereas our agents cannot be distinguished from each other (at least not at the beginning of the execution).

Model. We consider a variant of [11]'s ANTS problem, where a set of mobile *agents* search the infinite grid for an adversarially hidden treasure. The agents are controlled by asynchronous randomized finite state machines with a common sense of direction and 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 a designated grid cell referred to as the *origin* (say, the cell with coordinates $(0, 0) \in \mathbb{Z}^2$). We assume for simplicity that

the agents can distinguish between the origin and the other cells. We denote the cells with either x or y-coordinate being 0 as *north/east/south/west-axis*, depending on their location.

The main difference between our variation of the ANTS model and the original one lies in the agents' computation and communication capabilities. In both variants, all agents run the same (randomized) protocol, however, under the model considered in the present paper, the agents are controlled by an asynchronous randomized *finite state machine* (FSM). This means that the individual agent has a constant memory and thus, in general, can store neither coordinates in $\mathbb{Z}^2$ nor the number of agents. On the other hand, in contrast to the model considered in [11], our agents may communicate with each other. Specifically, under 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 quite limited though: agent a merely senses for each state q of the finite state machine, 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 [9] with bounding parameter $b = 1$.

The *distance* between two grid cells $(x, y), (x', y') \in \mathbb{Z}^2$ 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, S, E, 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.

The agents operate in an asynchronous environment. Each agent's execution progresses in discrete (asynchronous) steps indexed by the non-negative integers and we denote the time at which agent a completed step $i > 0$ by $t_a(i) > 0$. Following the common practice, we assume that the time stamps $t_a(i)$ 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.

Formally, the agents' protocol is captured by the 3-tuple $\Pi = \langle Q, s_0, \delta \rangle$, where Q is the finite set of *states*; $s_0 \in Q$ is the *initial state*; and

$$\delta : Q \times 2^Q \to 2^{Q \times \{N,S,E,W,P\}}$$

is the *transition function*. At time 0, all agents are in state s_0 and positioned in the origin. 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 a at time $t_a(i+1)$ and its corresponding position transition $\tau \in \{N, S, E, W, P\}$ are dictated based on the transition function δ by picking the pair $(q', \tau) \in \delta(q, Q_a)$, 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 $t_a(i)$. (Step i is deterministic if $|\delta(q, Q_a)| = 1$.) For simplicity, we assume

that while the state subset 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 $[t_a(i), t_a(i+1))$.

The goal of the agents is to locate an adversarially hidden *treasure*, i.e., to bring at least one agent to the cell in which the treasure is positioned. The distance of the treasure from the origin is denoted by D. As in [11], we measure the performance of a protocol in terms of its run-time, where the time is scaled so that $t_a(i+1) - t_a(i) \leq 1$ for every agent a and step $i \geq 0$. Although we express the run-time complexity in terms of the parameters n and D, we point out that neither of these two parameters is known to the agents (who cannot even store them in their very limited memory).

2 Parallel Rectangle Search

In this section, we introduce the collaborative search strategy RS (Rectangle-Search) that depends on an *emission scheme*, which divides all participating agents in the origin into *teams* of size ten and emits these teams continuously from the origin until all search teams have been emitted. We delay the description of our emission scheme until Section 3 and describe for now the general search strategy (without a concrete emission scheme). We assume, for the sake of the following informal explanation, an environment in which the agents operate in synchronous rounds and then explain how we can lift this assumption.

The RS strategy consists of two stages. The first stage works as follows: Whenever a team is emitted, one agent becomes an *explorer* and four agents become *guides*, one for each cardinal direction. The remaining five agents become *scouts*, whose function will be explained later. Now, each guide walks into its respective direction until it hits the first cell that is not occupied by another guide. The explorer follows the north-guide and when they hit the non-occupied cell $(0, d) \in \mathbb{Z}^2$ for some $d > 0$, the explorer starts a *rectangle search* by first walking south-west towards the west-guide. When it hits a guide, the explorer changes direction to south-east, then to north-east, and finally to north-west. This way, it traverses all cells in distance d from the origin, referred to hereafter as *level* d, (and also almost all cells in distance $d+1$). When the explorer meets a guide on its way, the guide enters a *sleep* state to be awoken again in the second stage. The explorer also enters a sleep state after arriving again at the north-guide, thereby completing the first stage of the rectangle search.

The second stage of RS is started when the last search team is emitted from the origin. At this point in time, $\Theta(n)$ cells are occupied by sleeping guides/explorers in all four cardinal directions. The last search team wakes up the innermost sleeping search team upon which it resumes its job and walks outwards to explore the next unexplored level in the same way as in the first stage. Each team recursively wakes up the search team of the next level until all sleeping teams have been woken up and resumed the search. A search in the second stage has one important difference in comparison to a search in the first

stage: When an explorer meets a guide g during a search, instead of entering a sleep state, g moves outwards to the next unexplored level, hopping over all the other stationary guides on its way, and waits there for the next appearance of an explorer. When the explorer has finished its rectangle by reaching the north-guide again, it moves north (with the north-guide) to the first unexplored level and starts another search there. Knowing that all other guides have reached their target positions in the same level as well, a new search can begin.

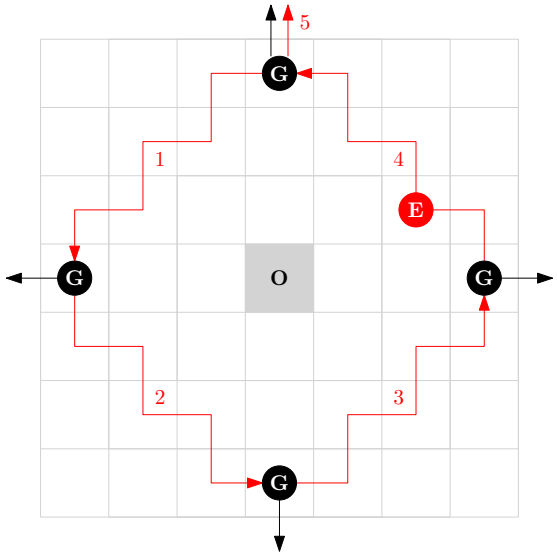

Fig. 1. An explorer (E) starts a rectangle search in level $\ell = 3$ at the north-guide, visits all guides (G) in level ℓ in a counter-clockwise fashion and ends at the north-guide. Whenever the explorer meets a guide, the guide moves outwards in its cardinal direction. When the explorer completes its search by arriving again at the north-guide, both agents walk outwards together to the next level to be searched. (The numbers indicate the order in which the explorer moves.)

Note that the (temporary) assumption of a synchronous environment is crucial for the correctness of the algorithm described so far as we assume that whenever an explorer crosses a coordinate axis, the respective cell contains a guide. In an asynchronous setting, the guide might still be on its way to that particular cell and hence, the explorer would continue walking diagonally ad infinitum. We counter this problem by coupling the searches for different levels in such a way that a search in level ℓ can never have progressed further than a search in level $\ell' < \ell$. This implies that a search in level ℓ cannot start/finish earlier than a search in level $\ell' < \ell$ starts/finishes. This coupling is implemented by equipping each explorer with a *scout* that essentially allows the explorer e_ℓ in level ℓ to check whether the explorer $e_{\ell-1}$ of the preceding level has already

progressed at least as far as e_ℓ and to move only then. On top of that, explorer e_ℓ only leaves a coordinate axis after ensuring, again by means of its scout, that there is already a guide present in level $\ell + 1$. This additional check (together with a few technicalities described later) suffices to ensure that the searches are "nested" properly and the corresponding guides of each explorer are waiting in the right positions along the coordinate axes when those are hit by the explorer.

As a (much desirable) byproduct of the aforementioned explorers' logic, it is guaranteed that during the execution of the RS strategy, every cell contains at most one explorer of each possible state. To ensure that the same holds for the guides, they are also equipped with scouts whose role is to check that during a guide's journey outwards, it does not move into a cell which is already occupied by a guide, unless the latter is in a stationary state (waiting for its explorer).

2.1 The RS strategy

Emission scheme. Initially, all n agents are located at the origin. Until all agents become involved in the RS strategy, an *emission scheme* is responsible for emitting new teams (each consisting of ten agents) from the origin. The emission of the teams is spaced apart in time in the sense that no two teams are emitted at the exact same time. (Under a synchronous schedule, a spacing of 20 time units is guaranteed.) To formally express (and analyze) the emission rate, we introduce the notion of an *emission function* $f_n : \mathbb{N}_0 \to \mathbb{N}_0$, where, until all teams are emitted, $f_n(t)$ bounds from below the number of teams emitted up to time t. For simplicity, we assume that there are enough agents to execute our algorithms, i.e., that $n \geq 30$.

Let $k + 2$, $k \in \Theta(n)$, be the total number of emitted teams where we assume $k \geq 2$. The first and last emitted teams have a special role as *signal teams* in our protocol. The k remaining teams $s_1, \ldots, s_k$ will be referred to as *search teams*. Whenever a search team becomes ready, four of the ten agents become MGuides — one for each cardinal direction — and walk outwards in their corresponding directions, while the fifth one becomes a MExp and follows the north-MGuide (see below for a detailed description of the agent types). Each MGuide and MExp is accompanied by a Scout that will stick to this particular agent for the rest of the execution.

Agent types. In the remainder of the paper, we will refer to several different types of agents. Since there is only a constant number of different types, these can be modeled by having individual finite automata for the various types. We essentially use six different types and explain their specific behavior in the following: Scout, Guide, MGuide (for moving guide), MExp (for moving explorer), WExp (for waiting explorer), and Exp (for explorer). We will use the terms "outwards" and "inwards" in the context of agents of the two Guide-types (recall that they are associated with a cardinal direction) to indicate the respective direction away from or towards the origin. We subsume the types Exp/MExp/WExp and Guide/MGuide under the name *explorers* and *guides*, resp. During the process of the algorithm, each non-Scout agent will be accompanied by a Scout, whose

type is specific to the type of the agent it is accompanying — its *owner*. Since all different Scout-types have very similar tasks, we first give a general description of a Scout's function and then explain its type-specific behavior together with the owner's behavior.

Scout. The function of each Scout-type is to control when its owner is allowed to move further. It does so by moving to one of the four neighbor cells of the owner – the *scout position* – and waiting for a certain condition (the presence/absence of a certain type of agent) to become true in that cell. When the condition is met, the Scout moves back to the cell containing its owner and notifies the owner. When the owner moves to a new position, the Scout moves along. As Scouts only play an auxiliary role in our protocol, we may refer to a cell as *empty* even if it contains Scouts.

Guide. A Guide waits until a Exp performing a search (this can be encoded in the state of the Exp) has entered its cell. When (i) the cell one coordinate inwards is empty and (ii) its cell contains neither MGuide nor MExp, it becomes a MGuide.

MGuide. A MGuide moves outwards (at least one cell) until it hits a cell c that contains no Guide. The north-MGuide moves north together with the MExp of the same search team. It does so by verifying before each move (after the first) that the MExp has caught up and is in the same cell. Otherwise, it waits for the MExp to catch up. Upon arriving in c, the MGuide becomes a Guide, and waits for an Exp to visit. A MGuide uses its Scout to prevent moving in a cell that contains a MGuide, MExp or Exp.

MExp. A MExp repeatedly moves north together with the north-MGuide of the same search team. More precisely, it only moves north when there is no MGuide in its cell, implying that the MGuide is already one cell further and waits there for the MExp to catch up. The MExp moves until it hits the first cell c that contains neither a Guide that already has an Exp searching (this can be encoded in the state of the Guide) nor an Exp. As soon as cell c contains a Guide (the north-Guide of this explorer's team), it becomes an Exp. A MExp uses its Scout to prevent moving into a cell that contains another MExp while walking outwards.

WExp. A WExp waits until its cell is empty and then becomes a MExp.

Exp. An Exp does the bulk of the actual search process by moving along the sides of a rectangle using Guides on its way to change direction. In the process, it moves south-west, then south-east, north-east, and north-west, in this order. Initially, an Exp performs one move west and then alternatingly south and west.

During a diagonal walk, an Exp uses its Scout to prevent it from overtaking Exps closer to the origin during their search as follows. Consider an Exp e in the north-west quarter-plane (walking south-west). The Scout is sent to the south-neighbor cell, referred to as the *scouting position*, and notifies e, when no Exp present there (which might immediately be the case). Only then, the Exp and the Scout move one cell further where the Scout again enters the scouting position.

When the Exp meets a west/south/east-Guide in an axis cell c, it changes its moving direction. Before leaving the axis, it waits until c does contain neither Guide nor MGuide (thereby ensuring that there is a Guide one cell outwards).

Upon arrival back at the north Guide after the rectangle search is completed, it becomes a WExp.

The Exp of the search team exploring level 1 counts its steps (the exploration journey at this level contains exactly 8 cells) and uses the Scout to make sure that the cells on the coordinate axes contain a Guide before entering them.

The signal teams. The first and last emitted teams, s_0 and s_{k+1}, resp., have a special role and they do not actively participate in the exploration of the grid (which is handled by $s_1, \ldots, s_k$). Their job is solely to signal to the other teams when the second stage of the protocol begins.

The first team s_0 enters a special *signal* state and stays at the origin until the last team s_{k+1} has been emitted. (Due to the design of our emission scheme in Section 3, the agents in team s_{k+1} know that they belong to the last emitted team and are able to notify the agents of s_0 accordingly.) The aforementioned logic of the agents in RS ensures that as long as there is an agent present in the origin, the Guides and Exps of the innermost search team (and recursively all other search teams) cannot move outwards. When s_0 is notified by s_{k+1}, the agents in both teams switch to a designated idle state, ignored by all other agents. As now the origin appears to be empty, the Guides and Exps of the innermost (and eventually the other search teams) can move outwards to continue searching — the second stage has begun.

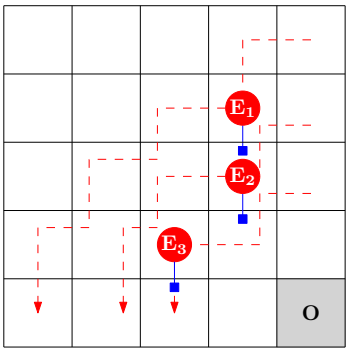

Fig. 2. Three Exps (red circles) are performing a search of adjacent levels in the northwest quadrant. Their Scouts are depicted as blue squares connected to their respective Exp. E_1 cannot walk further as there is still an Exp (E_2) in the cell checked by its Scout. Both E_2 and E_3 can walk further as their Scouts do not observe an explorer in the checked cells.

2.2 Correctness

In this section we establish the correctness of the RS strategy by proving that each cell is eventually explored and no agent is lost in the process. We say that a cell in level ℓ is *explored* after it has been visited by an Exp exploring level

ℓ, where we recall that level $\ell \in \mathbb{N}_0$ consists of all cells in distance ℓ from the origin. An Exp is said to *start* a (rectangle) search in level ℓ at time t if it moves west from the cell $(0, \ell)$ (containing the north Guide) at time t and it *finishes* a (rectangle) search in level ℓ at time t if it enters the cell $(0, \ell)$ from the east at time t. The *start time* t_ℓ^S, *finish time* t_ℓ^F, and *move time* t_ℓ^M are given by the times at which an Exp starts a search in level ℓ, finishes a search in level ℓ, and when the WExp in level ℓ becomes a MExp, resp. An Exp *explores* level ℓ at time t, if $t_\ell^S < t < t_\ell^F$. The design of RS ensures that regardless of the emission scheme used, the Guides in every cardinal direction occupy a contiguous segment of cells. It also implies the following observation and lemma.

Observation 1. *For two levels $\ell' > \ell$, we have $t_{\ell'}^S > t_\ell^S$, $t_{\ell'}^F > t_\ell^F$, and $t_{\ell'}^M > t_\ell^M$.*

Proof. Consider some level $\ell > 1$. An Exp can only start a search in level ℓ by leaving cell $(0, \ell)$ after the Exp of level $\ell - 1$ has left cell $(0, \ell - 1)$, which implies $t_\ell^S > t_{\ell-1}^S$. An Exp in level ℓ can only move to cell $(0, \ell)$ and thereby finish the search in level ℓ after the Exp of level $\ell - 1$ has already reached the cell $(0, \ell - 1)$, thus $t_\ell^F > t_{\ell-1}^F$. A WExp in level ℓ can become a MExp only when its Guide has left cell $(0, \ell)$. This requires in turn that the Exp in level $\ell - 1$ has already left cell $(0, \ell - 1)$ which requires that it already became a MExp, hence $t_\ell^M > t_{\ell-1}^M$. □

Lemma 2. *Outside the origin, no two agents of the same type occupy the same cell at the same time.*

Proof. First, recall that the MExps emitted from the origin are emitted at different times. A WExp becomes a MExp only when there is no other MExp in the same cell. MExps use Scouts to prevent stepping onto each others' cells. A MExp becomes an Exp in cell c only if c does not contain an Exp and Exps use Scouts to prevent stepping into each others' cells. As no two Exps can be in the same cell, neither can be two WExps and the claim for all explorer agents follows.

MGuides are emitted from the origin at different times. A Guide becomes a MGuide only if its cell does not contain a MGuide. MGuides use Scouts to prevent stepping onto each others' cells. A MGuide becomes a Guide only if its cell does not contain a Guide, which establishes the assertion for the guides.

As the Scouts of owners with different types also have different types, we only need to show the claim for Scouts of the same type. This follows by labeling a Scout as north/east/south/west-CScout, while it is checking a scout condition in the north/east/south/west neighboring cell of its owner's cell. □

Each Exp relies on Guides to indicate when it has to change the search direction in order to search a specific level. The next lemma gives a guarantee for this.

Lemma 3. *Whenever an Exp enters a cell c on an axis, cell c contains a Guide.*

Proof. Observe that if c lies on the north-axis, it will contain a Guide when the Exp e returns there because e only leaves c to start a search when c contains a Guide and this Guide stays there until e returns. To prove the claim for the other axis, let s_ℓ be the search team exploring level ℓ and let e_ℓ be the corresponding

explorer. We prove the statement by induction on ℓ. Observe that the statement holds for e_1 as the Exp of the first search team explicitly counts cells and uses its Scout to ensure that c contains a Guide.

Consider a cell c_ℓ on an axis in level ℓ and assume as the induction hypothesis that the cell $c_{\ell-1}$ on the axis in level $\ell-1$ contained a Guide when it was entered by Exp $e_{\ell-1}$. As $e_{\ell-1}$ blocks e_ℓ from overtaking it, Exp e_ℓ can enter c_ℓ only after $e_{\ell-1}$ has left $c_{\ell-1}$, which $e_{\ell-1}$ only does after ensuring that $c_{\ell-1}$ is empty. This requires, in turn, that the Guide in cell $c_{\ell-1}$ has become a MGuide and left $c_{\ell-1}$. This can only happen after all other MGuides have passed $c_{\ell-1}$ and the MGuide in cell c_ℓ has become a Guide, thereby asserting the claim. □

The canonical paths. In what follows, we use paths in the infinite grid in their usual graph-theoretic sense, viewing a path p as a (finite or infinite) sequence of cells, where $p(i)$ and $p(i+1)$ are grid neighbors for every $i \geq 1$. Notice that unless stated otherwise, the paths mentioned are not necessarily simple.

Let $s_1, \ldots, s_k$ be the search teams emitted from the origin (ignoring the two signal teams s_0 and s_{k+1}) ordered by ascending emission time and consider some agent a participating in one of the search teams $s_1, \ldots, s_k$. Given some adversarial policy ψ, let p_a^ψ be the path traversed by a during the execution of the algorithm under ψ starting at the time at which a is emitted from the origin. We extend the sequence defined by p_a^ψ, fixing $p_a^\psi(0) = (0,0)$. We shall refer to p_a^ψ as the *execution path* of a (under ψ).

The logic of the guides directly implies that if agent a is a north/south/east/west guide, then its execution path satisfies $p_a^\psi(i) = (0,i)/(0,-i)/(i,0)/(-i,0)$ for every adversarial policy ψ. In other words, the path traversed by a guide does not depend on the adversarial policy. We argue that this is in fact the case for all agent types, introducing the notion of a *canonical path*.

Lemma 4. *For every $1 \leq i \leq k$ and for each agent role ρ (among the 10 different roles in a search team), there exists a canonical path $p^*_{i,\rho}$ such that if agent a is the ρ-agent in search team s_i, then $p_a^\psi = p^*_{i,\rho}$, regardless of the adversarial policy ψ.*

Proof. As noted above, the assertion holds for the guides. Since the execution path of a scout is fully determined by the execution path of its owner, it suffices to show that the assertion holds for the explorer e_i of search team s_i.

Let $m_{i,j}$ be the simple path leading from cell $(0,i)$ to cell $(0,j-1)$ along the north axis and let r_ℓ be the simple path corresponding to the rectangle search of level ℓ, starting in cell $(0,\ell)$ and ending in cell $(1,\ell)$, that is, the path traversed by an explorer exploring level ℓ during the time interval $[t_\ell^S, t_\ell^F)$. We argue that the execution path of explorer e_i is always

$$p^*_{i,e} = m_{0,i} \circ r_i \circ m_{i,k+i} \circ r_{k+i} \circ m_{k+i,2k+i} \circ r_{2k+i} \cdots ,$$

namely, agent e_i (and search team s_i) search levels $\ell = z \cdot k + i$ for $z = 0, 1, \ldots$, where we recall that k is the number of search teams.

To establish this argument, we prove that if e_i currently searches level ℓ, then the next level it is going to search is $\ell+k$. The argument then follows by induction on z as the first level searched by e_i (after it is emitted from the origin) is level i. To that end, consider the explorer e_i at time t_ℓ^F when it is back in cell $(0,\ell)$ as an Exp. Recall that e_i becomes a MExp and starts moving outwards at time t_ℓ^M that occurs only after the corresponding north-Guide has become a MGuide and left cell $(0,\ell)$. This is turn happens only after cell $(0,\ell-1)$ was verified as empty, which implies that at time t_ℓ^M, every other MExp e_j is positioned in some cell $(0,\ell')$, $\ell' > \ell$. Since a MExp does not enter a cell containing another MExp, it follows that e_i will not overtake e_j as long as both are MExps, but rather pass over its north-Guide after e_j has already started exploring its next level. Therefore, explorer e_i will have to pass the north-Guides of all other search teams before it gets to its next explored level, which completes the proof as there are k search teams in total. □

It will sometimes be convenient to use the notation p_a^* for the canonical path $p_{i,\rho}^*$ when agent a is the ρ-agent of search team s_i. The key to Lemma 4's proof is the observation that since MExps do not overtake each other, the explorers maintain a *cyclic order* between them in terms of the levels they explore. The exact same argument can be applied to the guides, concluding that the agents of a search team "stick together" throughout the execution.

Corollary 5. *The agents that were emitted from the origin as guides of search team s_i serve as Guides in levels $\ell = z \cdot k + i$ for $z = 0, 1, \ldots$*

Preventing dead/live-locks. We now turn to prove that RS does not run into deadlocks. Recall that during the execution of RS, agents often wait for other agents to complete some task before they can proceed. In particular, we say that agent a is *delayed* by agent a' at time t, denoted $a \rightarrow_t a'$, if at time t, a is positioned in some cell c and resides in some state q and the RS strategy dictates that a can neither leave cell c nor move to any state other than q until a' performs some action in cell c that may take the form of entering cell c, leaving cell c, or moving to some state within cell c. For example, a guide in an axis cell c is delayed by its corresponding explorer until the latter reaches c. Another example is an explorer which is delayed by its scout in some north-west quarter-plane cell (x,y), while the latter is delayed until the explorer exploring the previous level leaves cell $(x,y-1)$. To avoid the necessity to account for the scouts, we extend the definition of delays in the context of the correctness proof, allowing for agent a in cell c to be delayed by agent a' in a neighboring cell c' if a is actually delayed by its scout in c who is delayed by a' in c'.

Let $\mathcal{D}_t$ be the directed graph that corresponds to the binary relation $\rightarrow_t$ over the set of agents. We prove that RS does not run into deadlocks by establishing the following lemma.

Lemma 6. *The directed graph $\mathcal{D}_t$ does not admit any (directed) cycle at all times t.*

Proof. Consider a snapshot of the agents' states and positions at time t. Examining the RS strategy, one realizes that the outermost MExp and MGuides are not delayed by any other agent and that the i^{th} outermost MExp and MGuides can only be delayed by the $(i-1)^{\text{th}}$ outermost MExp and MGuides. The innermost Exp e is not delayed by any agent as long as it is not in an axis cell. In an axis cell, e can only be delayed by the corresponding guide. An innermost guide in cell c is delayed by its corresponding explorer until the latter reaches cell c and since then, it can only be delayed by the corresponding innermost MGuide. Non-innermost Exp and Guides in level ℓ can only be delayed by the Exp and Guides in level $\ell-1$ or by the MExps and MGuides. The assertion follows. □

The following corollary is derived due to Lemma 6 since there is a constant number of state transitions an agent positioned in cell c can perform before it leaves cell c.

Corollary 7. *Agent a reaches cell $p_a^*(i)$ within finite time for every $i \geq 1$.*

Since the canonical path p_a^* contains infinitely many different nodes for every agent a, we can deduce from Corollary 7 that RS does not run into livelocks, thus establishing the following theorem.

Theorem 8. *The cell containing the treasure is explored in finite time.*

2.3 Runtime Analysis

For the sake of a clearer run-time analysis, we analyze RS employing an ideal emission scheme with emission function $f_n(t) = \Omega(t)$, i.e., a new search team is emitted from the origin every constant number of time units. We do not know how to implement such a scheme, but in Section 3, we will describe an emission scheme with an almost ideal emission function of $f_n(t) = \Omega(t - \log n)$ and in Section 4, we will show how to compensate for the gap.

Our proof consists of two parts. First, we analyze the run-time of RS assuming a "synchronous" adversarial policy ψ^{s}, where $t_a(i) = i$ for all a and i. Then, we lift this assumption by showing that ψ^{s} is actually the worst case policy. We start with the following lemmas.

Lemma 9. *Under ψ^s, we have $t^M_{\ell+1} - t^M_\ell \geq 4$ and $t^S_{\ell+1} - t^S_\ell \geq 4$.*

Proof. By Observation 1, we know that a search in level $\ell+1$ cannot finish before a search in level ℓ. By construction of the algorithm, $e_{\ell+1}$ cannot become a MExp in level $\ell+1$ before time $t^M_\ell + 4$, hence $t^M_{\ell+1} - t^M_\ell \geq 4$.

Consider the two explorers $e_{\ell+1}$ and e_ℓ exploring levels $\ell+1$ and ℓ, resp. The design of the emission process and the inequality $t^M_{\ell+1} - t^M_\ell \geq 4$ imply that when $e_{\ell+1}$ and e_ℓ were MExps, they had a distance of at least 3 with $e_{\ell+1}$ being closer to the origin. As $e_{\ell+1}$ has to walk to level $\ell+1$ to start a search and e_ℓ only to level ℓ, the claim holds. □

Lemma 10. *Under ψ^s, the explorer of search team s_i is not delayed after time t^M_i.*

Proof. Recall that t_i^M is the time when the Exp of search team s_i turns into a MExp after search level i in the first stage of the algorithm. By Lemma 9, we know that after time t_i^M, the distance between any two MExps at least 3 and hence, they cannot delay each other. It is easy to see that under ψ^{s}, an Exp never has to wait for the Guide of the next level in order to leave an axis and thus cannot be delayed by a Guide. Since $t_{\ell+1}^S - t_\ell^S \geq 4$, two Exps of adjacent levels can never delay each other either. □

Lemma 11. *Under ψ^s, we have $t_\ell^F \in \mathcal{O}(\ell + \ell^2/n)$ for any level $\ell > 0$.*

Proof. Consider level $\ell \leq k$ and recall that this level will be searched by the ℓ^{th} search team s_ℓ. Let t_ℓ be the time at which s_ℓ is emitted from the origin and note that f_n guarantees $t_\ell \in \mathcal{O}(\ell)$. Observe that no delays can occur during the first stage. Hence, s_ℓ reaches level ℓ after time $\mathcal{O}(\ell)$, visits the 8ℓ cells to explore level ℓ in time $\mathcal{O}(\ell)$ and thus guarantees $t_\ell^F \in \mathcal{O}(\ell)$. Moreover, since the last team is emitted from the origin by time $\mathcal{O}(k)$ and at this time, all search teams are positioned in the first k levels, it follows that each search team s_i starts its second stage by time $\mathcal{O}(k)$, that is, $t_i^M = \mathcal{O}(k)$.

Consider now level $\ell > k$. Assume wlog. that level ℓ is explored by search team i and let e be the explorer of that search team. Lemma 4 guarantees that e moves along its canonical path p_e^*. Let π be the canonical path p_e^* truncated after the exploration of level ℓ. Combining Lemma 10 with the fact that $t_i^M = \mathcal{O}(k)$, and recalling that $\ell > k = \Omega(n)$, it suffices to show that the length of π (in hops) is $|\pi| = \mathcal{O}(\ell^2/k)$.

To that end, we write $|\pi| = m_m + m_x$, where m_m is the number of hops in π that e performs as a MExp and m_x is the number of hops in π it performs as an Exp. The design of RS ensures that $m_m = \mathcal{O}(\ell)$ which is $\mathcal{O}(\ell^2/k)$ as $\ell > k$. Since e is part of s_i, we know that $m_x = \sum_{z=0}^{\lfloor \ell/k \rfloor} 8(i + zk) = \mathcal{O}(\ell^2/k)$, which yields the assertion. □

We now turn to show that the run-time of RS under any adversarial policy ψ is at most the run-time under ψ^{s}. By definition, policy ψ^{s} maximizes the length of the time between consecutive completion times of the agents' steps. Informally, we have to prove that by speeding up some agents, the adversary cannot cause larger delays later on.

To that end, consider two agents a and a' and recall that Lemma 4 guarantees that they follow the canonical paths p_a^* and $p_{a'}^*$, resp., regardless of the adversarial policy. The agents can delay each other only when they are in the same cell, so suppose that there exist two indices i and i' such that $p_a^*(i) = p_{a'}^*(i') = c$.

Given some adversarial policy ψ, let $t_{\text{in}}^\psi(a)$ (resp., $t_{\text{in}}^\psi(a')$) be the time at which agent a (resp., a') enters c in the step corresponding to $p_a^*(i)$ (resp., $p_{a'}^*(i')$) under ψ and let $t_{\text{out}}^\psi(a)$ (resp., $t_{\text{out}}^\psi(a')$) be the time at which agent a (resp., a') exits c for the first time following $t_{\text{in}}^\psi(a)$ (resp., $t_{\text{in}}^\psi(a')$) under ψ. The key observation now is that the adversarial policy does not affect the order in which a and a' enter/exit cell c.

Observation 12. *For every two adversarial policies ψ_1, ψ_2, we have $t_{in}^{\psi_1}(a) < t_{in}^{\psi_1}(a')$ if and only if $t_{in}^{\psi_2}(a) < t_{in}^{\psi_2}(a')$ and $t_{out}^{\psi_1}(a) < t_{out}^{\psi_1}(a')$ if and only if $t_{out}^{\psi_2}(a) < t_{out}^{\psi_2}(a')$.*

Therefore, the adversary may decide to modify its policy relatively to ψ^{s} by speeding up some steps of some agents, but this modification cannot delay the progression of the agents along their canonical paths. Corollary 13 now follows from Lemma 11.

Corollary 13. *Under any adversarial policy, $t_\ell^F \in \mathcal{O}(\ell + \ell^2/n)$ for any level $\ell > 0$.*

3 An Almost Optimal Emission Scheme

We introduce the emission scheme PTA (ParallelTeamAssignment) that w.h.p. guarantees an emission function of $f_n(t) = \Omega(t - \log n)$. In Section 4, we describe the search strategy GS (GeometricSearch), that yields an optimal run-time of $\mathcal{O}(D + D^2/n)$ when combined with RS. The main goal of this section is to establish the following theorem.

Theorem 14. *Employing the PTA emission scheme, RS locates the treasure in time $\mathcal{O}(D + D^2/n + \log n)$ w.h.p.*

Our first goal is to describe the process FS (FastSpread), where n agents spread out along the east ray R consisting of the cells $(x, 0)$ for $x \in \mathbb{N}_{>0}$ such that each cell in some prefix of R is eventually assigned to a unique agent. The main idea behind the implementation of FS is that on every step, agent a throws a fair coin and moves outwards (towards east) if the coin shows heads and stays put otherwise. If a senses that it is the only agent occupying cell c, then it marks itself as *ready* and stops moving; cell c is also said to be *ready* following this event. Furthermore, when a walks onto a ready cell, it moves outwards deterministically.

To prevent any cell from becoming empty, the agents employ a mechanism that ensures that at least one agent stays put in each cell. To implement this mechanism, the agents decide in advance, i.e., in step i, if they want to move in step $i+1$ and report their decision to the other agents. In other words, an agent a throws a coin in step i and enters a state H or T that correspond to throwing heads or tails, resp. Then, a moves outwards in step $i+1$ if and only if it entered state H in step i and if it senses at least one other agent in state T. Informally, a only moves if at least one other agent has promised to stay put next time it acts.

Next, we show that the protocol works correctly, i.e., no cell in the prefix of R will become empty before getting ready. Suppose for contradiction that there is a cell c, such that c becomes empty at time t. Let a be an agent and i a step of a such that for all agents a' in cell c and all steps j, it holds that $t_{a'}(j) \leq t_a(i) < t$. In other words, no agent in c changes its state during time

$t_a(i) < t' < t$. According to the design of our protocol, a must sense some other agent a' in state T precisely at time $t_a(i)$. Since a' does not wake up after $t_a(i)$ and before t, it follows that a' resides in state T at time t, which is a contradiction.

Lemma 15. *For every positive integer $s \leq 16n$, the first $s/16$ cells of the ray R are ready after $s + \mathcal{O}(\log n)$ time units w.h.p.*

Proof. Let X_a be the random variable that counts the number of moves a non-ready agent a made outwards by time $s \leq 16n$. Unfortunately, the moves that a makes are not independent of the previous moves. Therefore, we study a weaker probabilistic process, where the number of a's moves dominates X_a. Assume that a occupies cell c at time $t_a(j)$ and let $a_0, a_1, \ldots, a_{z-1}$ denote the set of non-ready agents that occupy cell c at time $t_a(j)$. In the weaker process, $a = a_i$ only moves in step $j + 1$ if a_{i+1} (index arithmetic in this proof is modulo z) is in state T and a is in state H at time $t_a(j)$ or if there is a ready agent in c.

Let j' be the last step a_{i+1} performs before $t_a(j)$. The probability that a_{i+1} enters state T in step j' is $1/2$. In addition, the probability that a_i enters H in step j is $1/2$ and therefore, the probability of a_i actually moving towards east in step $j+1$ is at least $1/4$ in the weaker process. Since we want to count movements of a_i that are independent of the previous movements, we divide the execution of FS into intervals of 2 time units. Now the last step that a_i executes in the end of each such interval only depends on the previous step that a_{i+1} executed. Since every agent wakes up at least once per time unit, both of these steps are unique for every interval and, in particular, independent of the previous time intervals.

Let X'_a be the random variable that counts the number of moves a made east in the weaker process conditioned on the event where a is not ready by time $s + \mathcal{O}(\log n)$. Since the coin tosses made in each step are independent and a moves towards east each 2 time units with probability at least $1/4$, we get that $E[X'_a] \geq (1/2) \cdot (1/4) \cdot (s + \mathcal{O}(\log n))$. By applying a Chernoff bound we get that $P(X'_a < 1/2 \cdot E[X'_a]) \in \mathcal{O}(n^{-h})$ for a choosable constant $h > 0$. Since $X_a \geq X'_a$, any agent a that is not ready by time $s + \mathcal{O}(\log n)$, the distance to the origin is at least $s/16$ w.h.p. □

Intuitively, the aforementioned process can be seen as parallel leader election. Since we want to describe an efficient emission scheme, it remains to show how the process can be used to quickly emit search teams consisting of five agents with their respective Scouts from the origin. To enable the FS procedure to elect ten different kinds of agents per search team, we dedicate every tenth cell to a specific kind of agent. As an example, every cell in distance $d \equiv 1 \pmod{10}$ is dedicated to an Exp. After an Exp is alone in a cell using the FS procedure described above, it collects its search team in the following manner: it first takes one step east where a leader election for the Scout dedicated to it takes place. If the corresponding cell is occupied by a Scout that is marked ready, they both move outwards to collect the next agent. Otherwise, the Exp waits until the leader election is over. After the Exp (accompanied by the collected Guides and

Scouts) collected all agents needed for the search team, the team walks to the origin from where it will then be emitted into the four cardinal directions. We refer to the FS protocol combined with the collection of the agents as PTA.

In addition, we always keep track of the innermost search team in the following way. We flag the agents in the leftmost cell that has not been collected as the innermost agents. Every time an agent moves out of the innermost cell with a coin toss, this flag is turned off. In addition, when the Exp collects its search team, it performs one additional move outwards to flag the cell where the Exp for the next team is elected as the innermost. An Exp only starts collecting its search team after it has been flagged as the innermost agent. This way we know that the closer the team is elected to the origin, the earlier it starts moving towards the origin. The use of the grid by the PTA is illustrated in Figure 3.

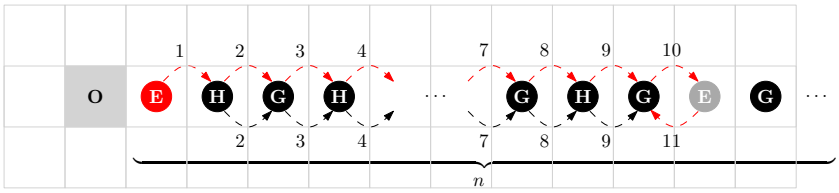

Fig. 3. First, the n agents executing the PTA protocol form a ray of single agents in cells $(0,1),\dots(0,n)$. After the innermost Exp e in cell $(0,1)$ (denoted with red circle) is ready, it starts collecting its search team. Assuming that all the agents to join its search team are ready, after at most ten time units, e has collected all the agents needed for its search team. Then in at most two more time units, e flags the Exp of the next team as innermost.

Similarly as in the RS protocol, every agent a always checks with its Scout before moving that the next cell is not occupied by an agent of the same type to prevent a from overtaking any of the other agents. Given that the agents never overtake or pass other agents, we observe that the canonical path p_a^* for any agent is fixed, i.e., is independent of the adversarial policy, starting from the time when a becomes ready. Furthermore, under ψ^{s}, all agents in a single team enter the origin simultaneously and PTA provides a spacing of more than 6 between emitted teams.

Lemma 16. *Assume that n agents start executing PTA protocol in round 0. Then at least $\lfloor \min\{s,n\}/10 \rfloor$ search teams have entered the origin by time $17s + \mathcal{O}(\log n)$ w.h.p.*

Proof. By Lemma 15, the first s cells are ready w.h.p. by time $t' = 16s+\mathcal{O}(\log n)$, which indicates that the agents in these cells are ready to perform their collection process latest at time t'. In addition, in each time unit after t', the Exps occupying one of the first s cells moves east unless they have already collected their teams. Therefore, latest at time $16s + \mathcal{O}(\log n) + 10 = t$, all full teams within the first s cells have been collected after which they start moving towards the origin.

Let $a_1, \ldots, a_m$ denote the non-Scout agents of some type, say Exps, within the first s cells. We first observe that after a_i has moved $10i$ times after being collected, it reaches the origin. Next, we point out that obeying a synchronous schedule, no agent ever gets blocked by the other agents. Assume that agent a_i is blocked by a_{i-1} from entering cell c. This indicates that a_i has made more than 1 move per time unit. Furthermore, a_i is able to move to c without blocks latest when it would have moved to c according to the synchronous schedule.

It follows that all agents from any team e_i reaches the origin latest when they would reach the origin according to the synchronous schedule. Since performing $10i$ moves takes $10i$ time units in the synchronous schedule, all agents in team e_m will reach the origin latest at time $t + 10m \leq t' + s \leq 17s + \mathcal{O}(\log n)$. □

By Lemma 16, the emission function $f_n(t)$ provided by the PTA protocol satisfies $f_n(t) = \Omega(t - \log n)$ and therefore, Theorem 14 follows.

4 Optimal Rectangle Search

In this section, we will present the search strategy *HybridSearch* that locates the treasure with optimal run-time of $\mathcal{O}(D + D^2/n)$. This is achieved by, combining RS employing the PTA with the randomized search strategy GS that is fast only if the treasure is close to the origin.

The search strategy GS is suited to locate the treasure very quickly if it is located close to the origin, more precisely if $D \leq \log(n)/2$. Initially, each of the n agents chooses uniformly at random one of the four quarter-planes that it will be searching. We will explain the strategy exemplary for an agent "responsible" for the north-east quarter-plane. The other three types operate analogously in their respective quarter-plane.

Initially, the agent moves one cell to the east. From then on, it moves a geometrically distributed number of steps east following which it moves a geometrically distributed number of steps to the north. More precisely, with probability $1/2$ the agent moves further and otherwise stops walking in the current direction. Both these processes can be realized in our model by having two state transitions where one of them moves the agent further while the other one ends the current walk. Either of the two transitions is chosen uniformly at random and a walk of geometrically distributed length is obtained.

Lemma 17. *If $D \leq \log(n)/2$, then GS locates the treasure in time $\mathcal{O}(D)$ w.h.p.*

Proof. Consider some cell c at distance $d \leq \log(n)/2$ from the origin and fix some agent a. Let X_a be a random variable that captures the length of the walk of agent a and observe that X_a obeys a negative binomial distribution so that

$$P(X_a = k) = (k+1) \cdot 2^{-(k+2)} \ .$$

Recalling that a has already moved one step, we conclude that the probability that a moves up to distance d is

$$P(X_a = d-1) = d \cdot 2^{-d-1} \geq 2^{-d-1} = \Omega(1/\sqrt{n}) \ .$$

Since all cells at distance d from the root have the same probability of being explored by a and since there are $\mathcal{O}(\log n)$ such cells, it follows that a explores cell c with probability at least $\Omega\left(\frac{1}{\sqrt{n}\log n}\right)$. Therefore, the probability that none of the agents explores cell c is at most

$$\left(1-\Omega\left(\frac{1}{\sqrt{n}\log n}\right)\right)^n < e^{-\Omega\left(\frac{\sqrt{n}}{\log n}\right)}.$$

The assertion follows. □

We can now combine the two search strategies GS, which is optimal for $D \leq \log(n)/2$, and RS employing PTA, which is optimal for $D = \Omega(\log n)$, into the HybridSearch strategy as follows.

At the beginning of the execution, each agent tosses a fair coin to decide whether it participates in RS or GS. Let n_r and n_g be the number of agents participating in RS and GS, resp. and observe that $n_r, n_g \geq n/3$ w.h.p. Then the agents enter according states so that they do not interfere with each other anymore. One group executes GS and locates the treasure w.h.p. in time $\mathcal{O}(D)$ if $D \leq \log(n)/2$ and the other group executes RS locates the treasure w.h.p. in time $\mathcal{O}(D + D^2/n)$ if $D = \Omega(\log n)$, thereby establishing Theorem 18.

Theorem 18. *HybridSearch locates the treasure in time $\mathcal{O}(D + D^2/n)$ w.h.p.*

References

1. Albers, S., Henzinger, M.: Exploring Unknown Environments. SICOMP, 2000
2. Aleliunas, R., Karp, R.M., Lipton, R.J., Lovasz, L., Rackoff, C.: Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems. In: SFCS, 1979
3. Alon, N., Avin, C., Koucky, M., Kozma, G., Lotker, Z., Tuttle, M.R.: Many Random Walks are Faster Than One. In: SPAA, 2008
4. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in Networks of Passively Mobile Finite-State Sensors. Distributed Computing, 2006
5. Aspnes, J., Ruppert, E.: An Introduction to Population Protocols. In: Middleware for Network Eccentric and Mobile Applications.
6. Baeza-Yates, R.A., Culberson, J.C., Rawlins, G.J.E.: Searching in the Plane. Information and Computation, 1993
7. Deng, X., Papadimitriou, C.: Exploring an Unknown Graph. JGT, 1999
8. Diks, K., Fraigniaud, P., Kranakis, E., Pelc, A.: Tree Exploration with Little Memory. Journal of Algorithms, 2004
9. Emek, Y., Wattenhofer, R.: Stone Age Distributed Computing. In: PODC, 2013
10. Feinerman, O., Korman, A.: Memory Lower Bounds for Randomized Collaborative Search and Implications for Biology. In: DISC, 2012
11. Feinerman, O., Korman, A., Lotker, Z., Sereni, J.S.: Collaborative Search on the Plane Without Communication. In: PODC, 2012
12. Fraigniaud, P., Ilcinkas, D., Peer, G., Pelc, A., Peleg, D.: Graph Exploration by a Finite Automaton. TCS, 2005
13. Förster, K.T., Wattenhofer, R.: Directed Graph Exploration. In: OPODIS, 2012

14. López-Ortiz, A., Sweet, G.: Parallel Searching on a Lattice. In: CCCG, 2001
15. Panaite, P., Pelc, A.: Exploring Unknown Undirected Graphs. In: SODA, 1998
16. Prabhakar, B., Dektar, K.N., Gordon, D.M.: The Regulation of Ant Colony Foraging Activity Without Spatial Information. PLoS Computational Biology, 2012
17. Reingold, O.: Undirected Connectivity in Log-Space. JACM, 2008