Trade-offs between Selection Complexity and Performance when Searching the Plane without Communication*

Christoph Lenzen
MIT
Cambridge, MA, USA
clenzen@csail.mit.edu

Nancy Lynch
MIT
Cambridge, MA, USA
lynch@csail.mit.edu

Calvin Newport
Georgetown University
Washington D.C., USA
cnewport@cs.georgetown.edu

Tsvetomira Radeva
MIT
Cambridge, MA, USA
radeva@csail.mit.edu

ABSTRACT

We argue that in the context of biology-inspired problems in computer science, in addition to studying the *time complexity* of solutions it is also important to study the *selection complexity*, a measure of how likely a given algorithmic strategy is to arise in nature. In this spirit, we propose a selection complexity metric χ for the ANTS problem [Feinerman et al.]. For algorithm $\mathcal{A}$, we define $\chi(\mathcal{A}) = b + \log \ell$, where b is the number of memory bits used by each agent and ℓ bounds the fineness of available probabilities (agents use probabilities of at least $1/2^\ell$).

We consider n agents searching for a target in the plane, at (unknown) distance at most D from the origin. We identify $\log \log D$ as a crucial threshold for our selection complexity metric. We prove a new upper bound that achieves near-optimal speed-up of $(D^2/n + D) \cdot 2^{\mathcal{O}(\ell)}$ for $\chi(\mathcal{A}) \leq 3 \log \log D + \mathcal{O}(1)$, which is asymptotically optimal if $\ell \in \mathcal{O}(1)$. By comparison, previous algorithms achieving similar speed-up require $\chi(\mathcal{A}) = \Omega(\log D)$. We show that this threshold is tight by proving that if $\chi(\mathcal{A}) < \log \log D - \omega(1)$, then with high probability the target is not found if each agent performs $D^{2-o(1)}$ moves. This constitutes a sizable gap to the straightforward $\Omega(D^2/n + D)$ lower bound in this setting.

Categories and Subject Descriptors

F.2.3 [**Analysis of Algorithms and Problem Complexity**]: Tradeoffs between Complexity Measures; G.3 [**Probability and Statistics**]: Markov processes; G.2.2 [**Graph Theory**]: Graph algorithms

General Terms

Algorithms, Theory

1. INTRODUCTION

It is increasingly accepted by biologists and computer scientists that the tools of distributed computation can improve our understanding of distributed biological processes. Examples include problems inspired by social insects: foraging for food [11, 12, 13], deciding on a location for a new nest [22], collectively carrying a large object [6]. A standard approach is to translate a biological process of interest (e.g., ant foraging [11, 13] or sensory organ pre-cursor selection [1]) into a formal problem in a distributed computing model, and then prove upper and lower bounds on the problem. The aim is to use these bounds to gain insight into the behavior of the motivating biological process.

A recognized pitfall of this approach is *incongruous analysis*, in which the theoretician focuses on metrics relevant to computation but not biology, or ignores metrics relevant to biology but not to computation. Motivated by this pitfall, this paper promotes the use of *selection complexity* metrics for studying biology-inspired distributed problems. Unlike standard metrics from computation, which tend to focus only on performance, selection complexity metrics instead attempt to measure the difficulty of a given algorithmic strategy arising in nature as the result of selective pressures. Roughly speaking, a solution with low selection complexity should be more likely to arise in nature than a solution with high selection complexity.

We argue that theoreticians studying biology-inspired problems should evaluate solutions in terms of selection complexity in addition to focusing on standard performance metrics; perhaps even measuring the trade-off between the two classes of metrics. This paper provides a case study of this approach by fixing a standard biology-inspired problem and new selection complexity metric, and then bounding the trade-off between performance and selection complexity with respect to this metric. In doing so, we also obtain results regarding concurrent non-uniform random walks that are of independent mathematical interest.

*This work is supported in part by AFOSR Contract Number FA9550-13-1-0042, NSF Award numbers: 0939370-CCF, CCF-1217506, CCF-AF-0937274, CCF 1320279, the DFG under reference number Le 3107/1-1., and the Ford University Research Program.

PODC'14, July 15–18, 2014, Paris, France.

ACM 978-1-4503-2944-6/14/07 ...$15.00.
http://dx.doi.org/10.1145/2611462.2611463 .

We recognize that most papers on biology-inspired distributed problems implicitly address selection complexity in their fixed model constraints. Restricting agents to not have access to communication in the search problem, for example, is a constraint that likely lowers the selection complexity of solutions in the model. What is new about our approach is that we are capturing such complexity in a variable metric, allowing us to study the trade-offs between algorithmic power and performance more generally. This can provide insights beyond those gained by characterizing the capabilities of a given static set of constraints.

In this paper, we focus on the problem of n probabilistic non-communicating agents collaboratively searching for a target in a two-dimensional grid placed at (unknown) distance D from the origin. For our lower bound, we assume that n is sub-exponential in D.[1] This problem is described and analyzed in recent work by Feinerman et al. [13] (referred to as the ANTS problem), who argue that it provides a good approximation of insect foraging and represents a useful intersection between biological behavior and distributed computation. The analysis in [13] focuses on the *speed-up* performance metric, which measures how the expected time to find the target improves with n. The authors describe and analyze search algorithms that closely approximate the straightforward $\Omega(D + D^2/n)$ lower bound for finding a target placed at distance D from the origin.

Selection Metric Motivation. We consider the selection complexity metric χ, which captures the bits of memory and probabilistic range used by a given algorithm. This metric is motivated by the fact that memory can be used to simulate small probability values, and such values give more power to algorithms, e.g. permitting longer directed walks with a given amount of memory. In more detail, for algorithm $\mathcal{A}$, we define $\chi(\mathcal{A}) = b + \log \ell$, where b is the number of bits of memory required by the algorithm and ℓ is the smallest value such that all probabilities used in $\mathcal{A}$ are bounded from below by $1/2^\ell$. In Section 3 and Section 4, we show that the choice of the selection metric arises naturally from the analysis of our algorithms and the lower bound.

We conjecture that, from a biological point of view, it is reasonable to assume that large values of ℓ are associated with higher selection complexity. Clearly, algorithms relying on small probabilities are more sensitive to additive disturbances of the probability values. Hence, creating a small probability based on a single event is harder to accomplish, since the event must not only have a strong bias towards one outcome, but also be well-protected against influencing factors (like temperature, noise, etc.). On the other hand, using multiple independent events to simulate one with larger bias (also known as probability boosting) constitutes a hidden cost. Our model and algorithms make this cost explicit, by accounting for it in terms of the memory needed for counting such events.

Results. In this paper, we generalize the problem of [13] by now also considering the selection complexity metric χ. We identify $\log \log D$, for target within distance D, as a crucial threshold for the χ metric when studying the achievable speed-up in the foraging problem. In more detail, our lower bound proves that for any algorithm $\mathcal{A}$ such that $\chi(\mathcal{A}) \leq \log \log D - \omega(1)$, there is a placement of the target within distance D such that the probability that $\mathcal{A}$ finds the target in fewer than $D^{2-o(1)}$ moves of each agent is polynomially small in D, and the probability of finding a target placed randomly within this distance is $o(1)$. The *speed-up* in this case is bounded from above by $\min\{n, D^{o(1)}\}$, as opposed to the optimal speed-up of $\min\{n, D\}$. At the core of our lower bound is a novel analysis of recurrence behavior of small Markov chains with probabilities of at least $1/2^\ell$.

Concerning upper bounds, we note that the foraging algorithms in [13] achieve near-optimal speed-up in n, but their selection complexity, as measured by $\chi(\mathcal{A})$, is higher than the $\log \log D$ threshold identified by our lower bound: these algorithms require sufficiently fine-grained probabilities and enough memory to randomly generate and store, respectively, coordinates up to distance at least D from the origin; this entails $\chi(\mathcal{A}) \geq \log D$. In this paper, we seek upper bounds that work for $\chi(\mathcal{A}) \approx \log \log D$, the minimum value for which good speed-up is possible. With this in mind, we begin by describing and analyzing a very simple algorithm that is non-uniform in D (agents know the value of D) and has asymptotically optimal expected running time. It illustrates our main ideas of walking up to certain points in the plane while counting approximately, thus using little memory, and showing that this is sufficient for searching the plane efficiently. This algorithm uses a value of $\chi = \log \log D + \mathcal{O}(1)$, which matches our lower bound result for χ up to factor $1 + o(1)$.

We generalize the ideas used in our simple algorithm to derive a solution that is uniform in D. The main idea is to start with some estimate of D and keep increasing it while executing a corresponding version of the simple search algorithm described above for each such estimate. Our uniform algorithm solves the problem after $\mathcal{O}(D^2/n + D) \cdot 2^{\mathcal{O}(\ell)}$ moves of each agent in expectation (if $\ell = \mathcal{O}(1)$, the algorithm matches the $\Omega(D^2/n + D)$ lower bound), for $\chi(\mathcal{A}) \leq 3 \log \log D + \mathcal{O}(1)$. We remark that the increased running time is due to the fact that in order to keep the value of χ small, we increase our estimate of D by a factor of $2^{\mathcal{O}(\ell)}$ each time, which may result in "overshooting" the correct distance by factor $2^{\mathcal{O}(\ell)}$. Note that this suboptimal expected running time arises from enforcing $o(\log \log D)$ memory bits; otherwise, one is always free to use only constant probabilities.

Discussion. An interesting question that arises from our results is the relationship between b and ℓ in the definition of $\chi(\mathcal{A})$: roughly speaking, more bits of memory might be of greater utility than having access to smaller probabilities. This seems intuitive given that smaller probability values can be simulated using additional memory (e.g., to simulate a coin that returns heads with probability $1/2^k$, flip a uniform coin k times while storing the number of coin tosses in the additional memory), but in general more precise probabilities cannot be used to simulate additional memory.

From a biological perspective, we do not claim that χ is necessarily the *right* selection metric to use in studying such problems. We chose it because b and ℓ seem to be important factors in search, and they are potentially difficult to increase in nature. However, we recognize that the refinement and validation of such metrics require close collaboration with evolutionary biologists. In this paper, our main goal is to advertise the selection complexity approach as a promising tool for studying biology-inspired problems.

From a mathematical perspective, we emphasize that our lower bound result, in particular, is of independent interest. It is known that uniform random walks do not provide

[1] Note that an exponential number of agents finds the target quickly even if they employ simple random walks.

substantial speed-up in the plane searching problem [3]; the speed-up is bounded by $\min\{\log n, D\}$. Our lower bound generalizes this observation from uniform random walks to probabilistic processes with bounded probabilities and small state complexities.

Related Work. This work was initially inspired by the results in [11] and [13], which originally introduced the problem studied here. More precisely, in [13] the authors present an algorithm to find the target in optimal expected time $\mathcal{O}(D^2/n + D)$, assuming that each agent in the algorithm knows the number of agents n (but not D). For unknown n, they show that for every constant $\epsilon > 0$, there exists a uniform search algorithm that is $\mathcal{O}(\log^{1+\epsilon} k)$-competitive, but there is no uniform search algorithm that is $\mathcal{O}(\log k)$-competitive. In [11], Feinerman et al. provide multiple lower bounds on the advice size (number of bits of information the ants are given prior to the search), which can be used to store the value n, some approximation of it, or any other information. In particular, they show that in order for an algorithm to be $\mathcal{O}(\log^{1-\epsilon} n)$-competitive, the ants need advice size of $\Omega(\log\log n)$ bits. Note that this result also implies a lower bound of $\Omega(\log\log n)$ bits on the total size of the memory of the ants, but only under the condition that optimal speed-up is required. Our lower bound is stronger in that we show that there is an exponential gap of $D^{1-o(1)}$ to the maximum speed-up if $\chi(\mathcal{A})$ is too small (and n is sub-exponential in D). Similarly, the algorithms in [13] require $\Omega(\log D)$ bits of memory, as contrasted with our algorithm that uses $b \leq 3\log\log D + \mathcal{O}(1)$ bits of memory.

Searching and exploration of various types of graphs by single and multiple agents are widely studied in the literature. Several works study the case of a single agent exploring directed graphs [2, 5, 7], undirected graphs [21, 23], or trees [8, 16]. Out of these, the following papers have restrictions on the memory used in the search: [16] uses $\mathcal{O}(\log n)$ bits to explore an n-node tree, [5] studies the power of a pebble placed on a vertex so that the vertex can later be identified, [8] shows that $\Omega(\log\log n)$ bits of memory are needed to explore some n-node trees, and [23] presents a log-space algorithm for st-connectivity. There have been works on graph exploration with multiple agents [3, 9, 15]; while [3] and [15] do not include any memory bounds, [9] presents an optimal algorithm for searching in a grid with constant memory and constant-sized messages in a model, introduced in [10], of very limited computation and communication capabilities. Note that even though these models restrict the agents' memory, the fact that the models allow communication makes it possible to simulate larger memory.

In the above papers, we have seen that the metrics typically considered by computer scientists in graph search algorithms are usually the amount of memory used and the running time. In contrast, biologists look at a much wider range of models and metrics, more closely related to the physical capabilities of the agents. For example, in [4] the focus is on the capabilities of foragers to learn about different new environments, [17] considers the physical fitness of agents and the abundance and quality of the food sources, [18] considers interesting navigational capabilities of ants and assumes no communication between them, [19] measures the efficiency of foraging in terms of the energy over time spent per agent, and [24] explores the use of different chemicals used by ants to communicate with one another.

2. MODEL

Our model is similar to the models considered in [11, 13]. We consider an infinite two-dimensional square grid with coordinates in $\mathbb{Z}^2$. The grid is to be explored by $n \in \mathbb{N}$ identical, non-communicating, probabilistic agents. Each agent is always located at a point on the grid. Agents can move in one of four directions, to one of the four adjacent grid points, but they have no information about their current location in the grid. Initially all agents are positioned at the origin. We also assume that an agent can return to the origin, and for the purposes of this paper, we assume this action is based on information provided by an oracle. We also assume the agent returns on a shortest path in the grid that keeps closest to the straight line connecting the origin to its current position. Note that the return path is no longer than the path of the agent away from the origin; therefore, since we are interested in asymptotic complexity, we ignore the lengths of the return paths in our analysis.

Agents. Each agent is modeled as a probabilistic finite state automaton; since agents are identical, so are their state automata. Each automaton is a tuple (S, s_0, δ), where S is a set of states, state $s_0 \in S$ is the unique starting state, and δ is a transition function $\delta : S \to \Pi$, where Π is a set of discrete probability distributions. Thus, δ maps each state $s \in S$ to a discrete probability distribution $\delta(s) = \pi_s$ on S, which denotes the probability of moving from state s to any other state in S. For our lower bound in Section 4, it is convenient to use a Markov chain representation of each agent. We can express each agent as a Markov chain with transition matrix P, such that for each $s_1, s_2 \in S$, $P[s_1][s_2] = \pi_{s_1}(s_2)$, and start state $s_0 \in S$.

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

Executions. An execution of a single agent is defined as a sequence $(s_0, s_1, \cdots)$ of states that an agent visits while executing some algorithm, where s_0 is the start state. An execution of an algorithm with n agents is just an n-tuple of executions of single agents. For our analysis of the lower bound, it is useful to assume a synchronous model. So, we define a *round* of an execution to consist of one transition of each agent in its Markov chain. Note that we do not use such synchrony for our algorithms. In order to consider probabilistic executions, note that the Markov chain (S, P) induces a probability distribution of executions in a natural way, by performing an independent random walk on S with transition probabilities given by P for each of the n agents.

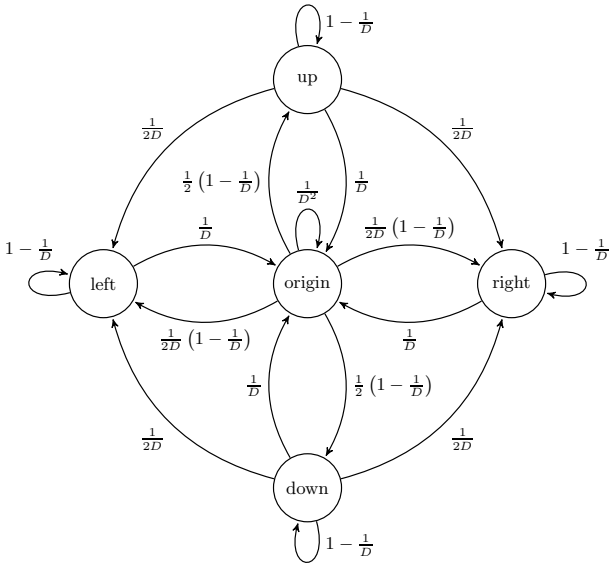
Problem Statement. The goal is to find a target located at some vertex at distance (measured in terms of the max-norm) at most D from the origin. Note that measuring paths in terms of the max-norm gives is a constant-factor approximation of the actual hop distance. We will consider both uniform and non-uniform algorithms with respect to D; that is, the agents may or may not know the value of D.

For simplicity, throughout this paper we will consider algorithms that are non-uniform in n, i.e., the agents' state machine depends on n.[2]

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

For our performance metric, we focus on the asymptotic running time in terms of D and n; more precisely, we are interested in the expected value of the metric M_{moves}: the minimum over all agents of the number of moves of the agent until it finds the target. We define the metric M_{steps} to be the minimum over all agents of the number of steps of the agent until it finds the target.

The selection metric of a state automaton (and thus a corresponding algorithm) is defined as $\chi(\mathcal{A}) = b + \log \ell$, where $b := \lceil \log |S| \rceil$ is the number of bits required to encode all states from S and $\ell \in \mathbb{N}$ is minimal with the property that $1/2^\ell \leq \min\{P[s,s'] \mid s,s' \in S \wedge P[s,s'] \neq 0\}$, the smallest non-zero probability value used by the algorithm.



State machine representation of Algorithm 1. State names match the values of the labeling function.

3. ALGORITHMS

In this section, we begin by describing a non-uniform algorithm in D that finds the target in asymptotically optimal time. The main purpose for presenting this algorithm is to illustrate our main techniques in a very simple setting. This algorithm uses probability values of the form $1/D$, which can easily be simulated using only biased coins that show heads with probability $1/2^\ell$ for any ℓ such that $\log D$ is an integer multiple of ℓ. We show that the target can be found in asymptotically optimal time using $b = \log\log D - \log \ell + 3$ bits of memory.

We then generalize this algorithm to work for the case of unknown D. This ensures that closer targets are found faster by the algorithm than targets that are far away. The way we achieve this is by starting with an estimate of D equal to 2 and repeatedly increasing it until the target is found. For each such estimate we execute the non-uniform algorithm. Since D is not known by the algorithm anymore, we cannot easily pick fixed values for some of the parameters we use in the algorithm in order to guarantee asymptotically optimal results for all possible values of D. Therefore, in our general algorithm the expected number of moves for the first agent to find the target becomes $(D^2/n + D) \cdot 2^{\mathcal{O}(\ell)}$ for $\chi = 3\log\log D + \mathcal{O}(1)$. Hence, for $\ell = \mathcal{O}(1)$ the algorithm is asymptotically optimal with respect to both metrics, and we achieve non-trivial speed-up of $\min\{n, D\}/D^{o(1)}$ for any $\ell \in o(\log D)$ (i.e., $\omega(1)$ bits of memory).

3.1 Non-uniform Algorithm

In this section we present an algorithm in which the value of D is available to the algorithm. We assume that $D > 1$; the cases of $D = 0$ and $D = 1$ are straightforward. Our general approach is the following: each agent chooses a vertical direction (up or down) with probability 1/2, walks in that direction for a random number of steps that depends on D, then does the same for the horizontal direction, and finally returns to the origin and repeats this process. We show that the minimum over all agents of the expected number of moves of the agent to find a target at distance up to D from the origin is at most $\mathcal{O}(D^2/n + D)$.

Let coin C_p denote a coin that shows tails with probability p. Using this convention, the pseudocode of this simple routine is given in Algorithm 1, accompanied by a state machine representation showing that the algorithm can be implemented using only three bits of memory. Later in this section we show that a slightly modified version of the algorithm guarantees that $\chi = \log\log D + 3$.

Algorithm 1: Non-uniform search.

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

We omit the complete proof of correctness of this simple algorithm, since it follows as a special case of the proof of our uniform algorithm in Section 3.2; for details we refer to the full version of the paper [20]. Below we state main performance complexity result and give a brief proof overview.

THEOREM 3.1. *Let each of n agents execute a copy of Algorithm 1. The minimum over all agents of the expected number of moves of an agent to find a target within distance $D > 1$ from the origin is $\mathcal{O}(D^2/n + D)$.*

[2]We remark that it is possible to apply a technique from [13] in order to generalize our results and obtain an algorithm that is uniform in both D and n.

Proof Sketch. First, we calculate the expected number of moves R for a single agent to complete an iteration of the main loop; by the properties of the geometric distribution $R \leq 2D$. Next, we note that the time to complete an iteration is not independent of whether the target is found or not, so we need to also compute the value of R conditioning on the event that the agent does *not* find the target in the given iteration; we call this value $\hat{R}$ and we can show that $\hat{R} \leq 2R$. Using the value of $\hat{R}$ and the fact that in the iteration in which an agent finds the target this takes at most $2D$ moves, we show that the expected number of moves $R_{i,a}$ for a fixed agent a to complete iteration i which is the earliest iteration in which the target is found is at most $4iD$.

Next, we move on to considering all n agents; we calculate the probability q that no agent finds the target after each agent executes one iteration of the main loop. Since the probability of a single agent to find the target in a given iteration is at least $1/(64D)$, we can bound $q \leq (1 - 1/(64D))^n \leq \max\{1 - \Omega(n/D), 1/2\}$ where the first value refers to the case $n \leq D$, and the second value refers to the case $n > D$.

Finally, we sum, over all $i \in \mathbb{N}$, the probability that some agent to find the target in iteration i (but no agent does so earlier) times the expected number of moves for an agent to finish iteration i conditioning on the event that this is the first iteration in which it finds the target. After some technical arguments about the right events to condition on in order to fix a specific agent that finds the target in each iteration i (see Theorem 3.10), we can show that this sum is at most $\sum_{i=1}^{\infty}(1-q)q^{i-1} \cdot \hat{R} = \mathcal{O}(D^2/n + D)$. □

We now generalize this algorithm to one that uses probabilities lower bounded by $1/2^\ell$ for some given $\ell \geq 1$. This is achieved by the following subroutine, which implements a coin that shows tails with probability $1/2^{k\ell}$ using a biased coin that shows tails with probability $1/2^\ell$, for $\ell \geq 1$.

Algorithm 2: coin(k, ℓ): Biased coin flip showing tails with probability $1/2^{k\ell}$.

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

LEMMA 3.2. *Algorithm 2 returns tails with probability* $1/2^{k\ell}$ *and requires* $\lceil \log k \rceil$ *bits of memory.*

Next, we combine Algorithm 1 and Algorithm 2, and we construct Algorithm Non-Uniform-Search by replacing the lines where coin $C_{1/D}$ is tossed in Algorithm 1 with a copy of Algorithm 2, with parameters $k = \lceil \log D/\ell \rceil$ and ℓ. Since Algorithm 2 does not generate any moves on the grid, the complexity bound in Theorem 3.1 holds.

THEOREM 3.3. *Let each of* n *agents execute a copy of Algorithm Non-Uniform-Search. The minimum over all agents of the number of moves to find a target in distance* $D > 1$ *from the origin is* $\mathcal{O}(D^2/n + D)$ *in expectation. Moreover, Algorithm Non-Uniform-Search satisfies that*

$$\chi(\textit{Algorithm Non-Uniform-Search}) = \log\log D + \mathcal{O}(1).$$

3.2 Uniform Algorithm

In this section, we generalize the results from Section 3.1 to derive an algorithm that is uniform in D. The main difference is that now each agent maintains an estimate of D that is increased until the target is found. For each estimate, an agent simply executes the corresponding variant of Algorithm Non-Uniform-Search. We show that for the algorithm in this section, the expected number of moves for the first agent to find a target at distance at most D from the origin is $(D^2/n + D)2^{\mathcal{O}(\ell)}$. Also, the algorithm uses only $b = 3\log\log D - 3\log\ell + \mathcal{O}(1)$ bits of memory.

To simplify the presentation, we break up the main algorithm into subroutines. We begin by showing how to move in a given direction by a random number of moves that depends on the current estimate $\hat{D}$ of D. In the following algorithm, recall that ℓ is used to bound from below the smallest probability available to each agent by $1/2^\ell$. We use an integer k as a parameter to the algorithm in order to generate different distance estimates $\hat{D} = 2^{k\ell}$.

Algorithm 3: walk(k,ℓ,dir): Move by a random number of moves in direction *dir* that is roughly uniform on $0, \ldots, 2^{k\ell}$.

```
while coin(k, ℓ) = heads do
    move one step in direction dir
```

LEMMA 3.4. *For each* $i \in \{0, \ldots, 2^{k\ell}\}$*, the probability that Algorithm 3 performs exactly* i *moves is at least* $1/2^{k\ell+2}$*. The probability that it performs at least* $2^{k\ell}$ *moves is at least* $1/4$*. The expected number of moves is smaller than* $2^{k\ell}$*. The algorithm requires* $\lceil \log k \rceil$ *bits of memory.*

This lemma follows from basic properties of the geometric distribution.

Using the subroutines above, Algorithm 4 visits each grid point of a square of side length $2^{k\ell}$ centered at the origin with probability $\Omega(1/2^{2k\ell})$.

Algorithm 4: search(k, ℓ): Visit each grid point of a square of side length $2^{k\ell}$ centered at the origin with probability $\Omega(1/2^{2k\ell})$.

```
if if C_{1/2} shows heads then
    walk(k, ℓ, up)
else
    walk(k, ℓ, down)
if C_{1/2} shows heads then
    walk(k, ℓ, right)
else
    walk(k, ℓ, left)
```

LEMMA 3.5. *If called at the origin, for each point* $(x, y) \in \{0, \ldots, 2^{k\ell}\}^2$*, Algorithm 4 visits* (x, y) *with probability at least* $1/2^{k\ell+6}$*. It can be implemented using* $\lceil \log k \rceil + 2$ *bits of memory.*

This follows from the observation that to move to a specific point (x, y) in the grid an agent needs (i) two specific outcomes of the unbiased coins (probability $1/2$ each), (ii) exactly $|x|$ steps up/down (probability $1/2^{k\ell+2}$ by Lemma

3.4), and (iii) at least $|y|$ steps left/right (probability 1/4 by Lemma 3.4).

Finally, in Algorithm 5, we use Algorithm 4 to efficiently search an area of $\mathcal{O}(D^2)$ with n agents. Intuitively, the algorithm iterates through different values of the outer-loop parameter i, which correspond to the different estimates of D, increasing by approximately a factor of 2^ℓ. For each such estimate, the algorithm needs to execute a number of calls to the search subroutine with parameter i. However, since agents have limited memory and limited probability values, we can only count the number of such calls to the search routine approximately. We do so similarly to Algorithm 3, by repeatedly tossing a biased coin and calling the search algorithm as long as the coin shows heads.

Algorithm 5: Search Algorithm for n agents. K is a sufficiently large constant.

```
for i = 1, ... do
    while coin(K + max{i − ⌊(log n)/ℓ⌋, 0}, ℓ) = heads
    do
        search(i, ℓ)
        return to the origin
```

Throughout the proof of the following results, we refer to an iteration of the outer-most loop as a phase.

Proof Overview. First, in Lemma 3.6, we calculate the expected number of moves R_i for an agent to complete phase i. Then, we determine the expected number of moves $\tilde{R}_{i,a}$ for an agent a to complete phase i (past some initial number of $\lceil \log_{2^\ell} D \rceil$ phases), conditioning on agent a finding the target in phase i, but not earlier. Next, we move on to reasoning about all n agents, instead of a single agent. In Lemma 3.8, we bound the probability that in each phase i, at least $\Omega(2^{i\ell})$ calls to the subroutine search(i, ℓ) are executed by all agents together. In Lemma 3.9, we use this result to calculate the probability that at least one of the n agents finds the target in some phase i. Finally, we use these intermediate results to prove the main result of this section, Theorem 3.10, which shows that the expected number of moves for the first agent to find a target within distance D from the origin is $2^{\mathcal{O}(\ell)}(D + D^2/n)$.

Denote by R_i the expected number of moves until an agent completes phase i. Let $\rho_i := 2^{(K+\max\{i-\lfloor(\log n)/\ell\rfloor,0\})\ell}$.

LEMMA 3.6. $R_i \leq 4\rho_i 2^{i\ell}$.

PROOF. Using linearity of expectation and independence of coin flips, R_i is the nested sum over all phases $i' \leq i$, the number j of calls to the search subroutine times the probability that j such calls are executed, and the number of moves k for an agent to complete a given call to the search subroutine times the respective probability: $\sum_{i'=1}^{i}(\sum_{j=0}^{\infty} 1/\rho_{i'}(1-1/\rho_{i'})^j \sum_{k=0}^{\infty} 1/2^{i'\ell}(1-1/2^{i'\ell})^k \cdot 2k) < 4\rho_i 2^{i\ell}$. □

Denote by $\tilde{R}_{i,a}$ the expected number of moves until a given agent a finds the target, conditioning on the fact that agent a finds the target in phase $i \in \mathbb{N}$, but no earlier phase.

COROLLARY 3.7. *If $i \geq i_0 = \lceil \log_{2^\ell} D \rceil$, then it holds that $\tilde{R}_{i,a} \leq 8\rho_i 2^{i\ell} + 2D = 2^{\max\{2i\ell - \log n, i\ell\}} \cdot 2^{\mathcal{O}(\ell)}$.*

PROOF. In the (first) call to the search subroutine in which agent a finds the target, it walks directly to the target, which takes at most $2D$ moves. Hence[3] $\tilde{R}_{i,a} \leq \tilde{R}'_{i,a} + 2D$, where $\tilde{R}'_{i,a}$ is the random variable counting all moves of a up to and including phase i, conditioning on a *not* finding the target in any of these phases. By the pseudocode, it is clear that the probability for an agent to miss the target in a given call to the search subroutine is at least 1/2. We partition the probability space according to the events that (1) the target is found during a given call to search, and (2) the target is not found during a given call to search. From the law of total expectation, it follows that $R_i \leq 1/2 \cdot \tilde{R}'_{i,a}$, so $\tilde{R}_{i,a} \leq \tilde{R}'_{i,a} + 2D \leq 2R_i + 2D$. The claim now follows from Lemma 3.6 and the fact that $2^{i\ell} \geq 2^{i_0\ell} \geq D$. □

[3]This inequality is to be read as probabilistic domination.

In the following lemmas and theorems, we switch from considering a probability distribution for one agent to considering a probability distribution for all n agents. Denote by $\mathcal{E}_1(i)$ the event that in total at least $2^{(K/2+i)\ell}$ calls to search(i, ℓ) are executed in phase i.

LEMMA 3.8. $P[\mathcal{E}_1(i)] \geq 1 - 1/2^{2\ell+2}$.

PROOF. By Lemma 3.2 and linearity of expectation, the expected number of calls to search(ℓ, i) performed by all agents during phase i is:

$$n \sum_{j=1}^{\infty} 1/\rho_i \cdot (1-1/\rho_i)^j \cdot j \geq n/2 \cdot \rho_i \geq 2^{(K+i-1)\ell}.$$

Since the coin flips are independent, we can apply Chernoff's bound to show that the probability that fewer than $2^{(K/2+i)\ell}$ searches are executed in total is at most $e^{-\Omega(K\ell)}$. The claim follows since K is a sufficiently large constant. □

Denote by $\mathcal{E}_2(i)$ the event that the target is found by some agent in phase i.

LEMMA 3.9. *For $i \geq i_0$, $P[\mathcal{E}_2(i)] \geq 1 - 1/2^{2\ell+1}$.*

PROOF. By Lemma 3.8, with probability at least $1 - 1/2^{2\ell+2}$, at least $2^{(K/2+i)\ell}$ iterations of the while loop are executed in total. Because $i \geq i_0 \geq \log_{2^\ell} D$, i.e., $2^{i\ell} \geq D$, Lemma 3.5 shows that in each iteration, the probability to find the target is at least $1/2^{i\ell+6}$. Therefore, the probability to miss the target in all calls is at most $(1 - 1/2^{i\ell+6})^{2^{(K/2+i)\ell}} = 2^{-\Omega(K\ell)}$.

Because K is sufficiently large, we may assume that this is at most $1/2^{2\ell+2}$. We infer that $P[\mathcal{E}_2(i)] \geq P[\mathcal{E}_2(i) \mid \mathcal{E}_1(i)] \cdot P[\mathcal{E}_1(i)] \geq (1 - 1/2^{2\ell+2})^2 \geq 1 - 1/2^{2\ell+1}$. □

Let event $\mathcal{E}_3(i)$ denote the event that the target is found for the first time in phase i.

THEOREM 3.10. *Let each of n agents execute Algorithm 5. The minimum over all agents of the expected number of moves for an agent to find a target within distance D from the origin is $2^{\mathcal{O}(\ell)}(D + D^2/n)$.*

PROOF. Observe that because the probability to find the target in phase i is independent of all coin flips in earlier phases, by Lemma 3.9 it follows that for $i \geq i_0$: $P[\mathcal{E}_3(i)] \leq \prod_{i'=i_0}^{i-1}(1 - P[\mathcal{E}_2(i')]) \leq 1/2^{(2\ell+1)(i-i_0)}$.

Let random variable X denote the number of moves until the first agent finds the target.

$$\mathbb{E}[X_{\text{found}}] = \sum_{i=1}^{\infty} P[\mathcal{E}_3(i)] \cdot \mathbb{E}[X_{\text{found}} | \mathcal{E}_3(i)].$$

Fix an arbitrary order of the agents (independently of the execution). We partition event $\mathcal{E}_3(i)$ into disjoint events $\mathcal{E}_3(i,a)$, where $\mathcal{E}_3(i,a)$ denotes the event that agent a is the minimal agent w.r.t. the chosen order that finds the target in phase i (by the definition of $\mathcal{E}_3(i)$, we know that such an agent exists). By the law of total expectation applied to the partition of event $\mathcal{E}_3(i)$, it follows: $\mathbb{E}[X_{\text{found}}|\mathcal{E}_3(i)] = \sum_a P[\mathcal{E}_3(i,a)|\mathcal{E}_3(i)] \cdot \mathbb{E}[X_{\text{found}}|\mathcal{E}_3(i,a)]$.

Let random variable $X_{i,a}$ denote the number of moves agent a takes to complete iteration i. Note that because we condition on event $\mathcal{E}_3(i,a)$, we know that the expected number of moves until the target is found by *some* agent is at most the expected number of moves for the fixed agent a to find the target and complete iteration i. Therefore, it follows that $\mathbb{E}[X_{\text{found}}|\mathcal{E}_3(i,a)] \leq \mathbb{E}[X_{i,a}|\mathcal{E}_3(i,a)]$.

By definition, we have $\mathbb{E}[X_{i,a}|\mathcal{E}_3(i,a)] = \tilde{R}_{i,a}$. We conclude that $\mathbb{E}[X_{\text{found}}|\mathcal{E}_3(i)] \leq \sum_a P[\mathcal{E}_3(i,a)|\mathcal{E}_3(i)]\tilde{R}_{i,a} = \tilde{R}_{i,a}$.

Finally, we sum over all phases to calculate the value of $\mathbb{E}[X_{\text{found}}]$. Using Corollary 3.7, we bound $\mathbb{E}[X_{\text{found}}]$ by.

$$\sum_{i=1}^{i_0} P[\mathcal{E}_3(i)] \cdot \tilde{R}_{i,a} + \sum_{i=i_0+1}^{\infty} P[\mathcal{E}_3(i)] \cdot \tilde{R}_{i,a}$$
$$\leq \tilde{R}_{i_0,a} + \sum_{i=i_0+1}^{\infty} \frac{1}{2^{(2\ell+1)(i-i_0)}} \cdot 2^{\max\{2i\ell - \log n, i\ell\}} \cdot 2^{\mathcal{O}(\ell)}.$$

Simple calculations bound this by $\max\{D^2/n, D\} \cdot 2^{\mathcal{O}(\ell)}$. □

4. LOWER BOUND

In this section, we present a lower bound showing that any algorithm $\mathcal{A}$ satisfying $\chi(\mathcal{A}) \leq \log\log D - \omega(1)$ does with high probability (w.h.p.) *not* find a target placed adversarially within distance D from the origin in $D^{2-o(1)}$ rounds. For uniform placement, this holds with probability $1 - o(1)$.

Throughout this section, we say that some event occurs with high probability iff the probability of the event occurring is at least $1 - 1/D^c$ for an arbitrary predefined constant $c > 0$ and some $D \in \mathbb{N}$. We say that two probability distributions π_1 and π_2 are *approximately equivalent* iff $\|\pi_1 - \pi_2\| = \mathcal{O}(1/D^c)$ for an arbitrary predefined constant $c > 0$. By $\|\cdot\|$ we denote the ∞-norm on the respective space.

First, we state the main theorem of the section in terms of the performance metric M_{steps}. Note that, by the definition of a round, this is equivalent to counting the expected number of rounds until the first agent finds the target. At the end of the section, in Corollary 4.10, we generalize the main result to apply to metric M_{moves}.

THEOREM 4.1. *Let $\mathcal{A}$ be an algorithm with $\chi(\mathcal{A}) = b + \log\ell \leq \log\log D - \omega(1)$ and $n \in 2^{D^{o(1)}}$ agents. There is a placement of the target within distance D from the origin such that w.h.p. no agent executing algorithm $\mathcal{A}$ finds it in fewer than $D^{2-o(1)}$ rounds. Moreover, the probability for some agent to find a target, placed uniformly at random in the square of side $2D$ centered at the origin, within $D^{2-o(1)}$ rounds is $o(1)$.*

4.1 Proof Overview

Here we provide a high-level overview of our main proof argument. We fix an algorithm $\mathcal{A}$ and focus on executions of this algorithm of $D^{2-o(1)}$ rounds. We prove that since agents have $o(\log D)$ states, they "forget" about past events too fast to behave substantially different from a biased random walk.

More concretely, first we show, in Corollary 4.3, that after $D^{o(1)}$ initial rounds each agent a is located in some recurrent class $C(a)$ of the Markov chain. We use this corollary to prove, in Corollary 4.4, that after the initial $D^{o(1)}$ rounds each agent a does not return to the origin (or it keeps returning every $D^{o(1)}$ rounds, so it does not explore much of the grid). Therefore, throughout the rest of the proof we can essentially ignore the states labeled "origin".

Assume there is a unique stationary distribution of $C(a)$. Since there are few states and non-zero transition probabilities are bounded from below, standard results on Markov chains imply that taking $D^{o(1)}$ steps from any state in the recurrent class will result in a distribution on the class's states that is (almost) indistinguishable from the stationary distribution (Corollary 4.5); in other words, any information agents try to preserve in their state is lost quickly with respect to D.

Next, we split up the rounds in the execution into groups such that within each group, rounds are sufficiently far apart from one another for the above "forgetting" to take place. For each group, we show that drawing states independently from the stationary distribution introduces only a negligible error (Lemma 4.6 and Corollary 4.7). Doing so, we can apply Chernoff's bound to each group, yielding that agents will not deviate substantially from the expected path they take when, in each round, they draw a state according to the stationary distribution and execute the corresponding move on the grid (Lemma 4.8 and Corollary 4.9). Taking a union bound over all groups, it follows that, w.h.p., each agent will not deviate from a straight line (the expected path associated with its recurrent class) by more than distance $o(D/|S|)$, where S is the number of states of the Markov chain. It is crucial here that the corresponding region in the grid, restricted to distance D from the origin, has size $o(D^2/|S|)$ and depends only on the agent's recurrent class. Therefore, since there are no more than $|S|$ components, taking a union bound over all agents shows that w.h.p. together they visit an area of $o(D^2)$.

4.2 Proof

W.l.o.g., we assume that values like $\ln D$ are integers; for the general case, one may simply round up. Moreover, since we are interested in asymptotics with respect to D, we may always assume that D is larger than any given constant.

Fix any algorithm $\mathcal{A}$, some $D \in \mathbb{N}$, and let $b + \log\ell \leq \log\log D - \omega(1)$. Consider the probability distribution of executions of $\mathcal{A}$ of length $\Delta = D^{2-o(1)}$ rounds; we will fix the $o(1)$-term in the exponent later, in Lemma 4.8.

We break the proof down into three main parts. First, in Section 4.2.1, we show that after a certain number of initial rounds each agent is in a recurrent class and, for simplicity, we can ignore the states labeled "origin". Next, in Section 4.2.2, we show that if we break down the execution into large enough blocks of rounds, we can assume that the steps associated with rounds in different blocks do not depend on each other, at marginal error. Finally, in Section 4.2.3, we focus on the movement of the agents in the grid, derived from these "almost" independent steps, and show that w.h.p. the agents will not explore any points outside of an area of size $o(D^2)$ around the origin.

4.2.1 Initial steps in the Markov chain

Let random variable $C(a, r)$ denote the recurrent class of the Markov chain in which agent a is located at the end of round r; if a is in a transient state at the end of round r, we set $C(a, r) := \perp$. Also, by p_0 we denote the smallest non-zero probability in the Markov chain. By assumption, we know that $p_0 \geq 1/2^\ell$.

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

Let $R_0 = p_0^{-2^b} 2^b c \log D = D^{o(1)}$ where the constant $c > 0$ will be specified later.

LEMMA 4.2. *For some agent a, round r, state s, condition on the event that during rounds $r, \ldots, r + R_0$ agent a never visits a state s' such that s is not reachable from s'. Then, w.h.p. a visits s in some round $r' \in \{r, \ldots, r + R_0\}$.*

PROOF. Since state s remains reachable, there must always be a path of length at most $|S| - 1$ from the state in which agent a is located in round $r' \in \{r, \ldots, r + R_0\}$ to state s. Therefore, the probability that the agent visits state s within R_0 rounds is bounded from below by the probability that a (biased) random walk on a line of $2^b \geq |S| - 1$ nodes starting at the leftmost node reaches the rightmost node within R_0 rounds. This probability in turn is bounded from below by $1 - (1 - p_0^{2^b})^{R_0/2^b} = 1 - (1 - p_0^{2^b})^{p_0^{-2^b} c \log D} \geq 1 - 1/2^{\Omega(c \log(D+n))} = 1 - 1/D^{\Omega(c)}$, where the second last step uses that $p_0 \leq 1/2$; otherwise, each state in the Markov chain has only one outgoing transition, in which case the claim of the lemma is trivial. Therefore, for an appropriate choice of c, w.h.p. agent a visits state s within R_0 rounds. □

In the following corollary we show that for a given agent and any round $r \geq R_0$, w.h.p. the agent is located in some recurrent class of the Markov chain.

COROLLARY 4.3. *For any agent a and any round $r \geq R_0$, w.h.p. $C(a, r) = C(a, r + 1) \neq \perp$.*

Follows from the fact that from any state in the Markov chain, there is always a reachable recurrent state. By Lemma 4.2, w.h.p. the agent reaches such a recurrent state in $D^{o(1)}$ rounds and never subsequently leaves the associated recurrent class.

Since the recurrent class $C(a, r)$ in which agent a is located is the same for all rounds $r \geq R_0$, we will refer to it by $C(a)$. Next, we show that the recurrent class $C(a)$ in which agent a is located does not contain any states labeled "origin", or otherwise, the agent keeps returning to the origin too often and makes no progress exploring the grid.

COROLLARY 4.4. *W.h.p., at least one of the following is true for any agent a and each round $R_0 \leq r \leq \Delta$: (1) agent a never visits a point in the grid at distance more than $D^{o(1)}$ from the origin, or (2) agent a is located in a recurrent class in which none of the states are labeled "origin".*

The idea of the proof is that if $C(a)$ contains a state labeled "origin", then we can show, by applying Lemma 4.2 to that state, that w.h.p. the agent visits that state every $D^{o(1)}$ rounds.

Throughout the rest of the proof, we consider executions after round R_0; since, $R_0 = D^{o(1)}$ and we consider executions of length $\Delta = D^{2-o(1)}$, we can just ignore these initial rounds. Therefore, from Corollary 4.3 and Corollary 4.4, we can assume for the rest of the proof that each agent a is in a recurrent class $C(a)$ and it does not return to the origin.

4.2.2 Moves drawn from the stationary distribution

Fix an agent a and consider $C := C(a)$. For the rest of the proof, we assume that C is aperiodic, and thus, has a unique stationary distribution π. In the general case, C may be periodic but we can apply standard results from Markov chain theory [14, Chapter XV.9] to show that by considering the Markov chain induced by P^t, where P is the probability matrix of the original Markov chain and t is its period, then a unique stationary distribution exists for some subset of states of C. The complete proof of the general case is available in the full version of the paper [20].

Consider blocks of rounds of size $\beta = c|S| \ln D / p_0^{|S|} = D^{o(1)}$, where $c > 0$ is a sufficiently large constant. We define groups of rounds such that each group contains one round from each block. Formally, for $1 \leq i \leq \beta$ and $j \in \mathbb{N}_0$, group B_i contains round numbers $i + j\beta \leq \Delta$.

Let $\pi_{r+\beta,s}$ denote the probability distribution on C of possible states agent a may be in at the end of round $r + \beta$, conditional on its state being $s \in C$ at the end of round r. Note that this distribution is, in fact, independent of r. We obtain the following corollary of Lemma 2 in [25] applied to the Markov chain restricted to class C.

COROLLARY 4.5. *There is a unique stationary distribution π of the Markov chain on C. For any state $s \in C$ and any round r, $\pi_{r+\beta,s}$ and π are approximately equivalent.*

From this corollary, we know that probability distributions π and $\pi_{r+\beta,s}$ are close to each other. Next, we identify the probability distribution that constitutes the "gap" between π and $\pi_{r+\beta,s}$.

LEMMA 4.6. *Let $1 \leq i \leq \beta$. Then, for any state $s \in C$ and any constant $c > 0$, there exists a probability distribution π_s such that $\forall r \in B_i, r \leq \Delta - \beta : \ \pi_s/D^c + (1 - 1/D^c)\pi = \pi_{r+\beta,s}$, where π is the unique stationary distribution of C.*

PROOF SKETCH. First, we bound the value of $\pi(s')$ for each $s' \in C$. We can show that $\pi(s') \geq p_0^{|S|}$ because p_0 is the minimum transition probability of getting to state s' from any other state, and we can do this at most $|S|$ times. Since $\ell + \log b \leq \log \log D - \omega(1)$, it follows that $p_0^{|S|} \geq 1/D^{o(1)}$. This also implies that $\pi(s') \leq 1 - 1/D^{o(1)}$ (assuming $|C| > 1$). Using the above equation involving π_s as definition, we can show that π_s is a probability distribution, i.e., for each $s' \in C$, $0 \leq \pi_s(s') \leq 1$ and $\sum_{s' \in C} \pi_s(s') = 1$. Here, we leverage that by Corollary 4.5 π and $\pi_{r+\beta,s}$ are approximately equivalent. □

We now show that within each class B_i, approximating the random walk of an agent in the Markov chain by drawing its state for each round $r \in B_i$ *independently* from the stationary distribution π does not introduce a substantial error. To this end, consider the following random experiment E_i. For each round $r + \beta \leq \Delta$, $r \in B_i$, we toss an independent biased coin such that it shows head w.h.p. In this case, we draw the state at the end of round r independently from π. Otherwise, we draw it from the distribution π_s, where s is the state the agent was in β steps ago and π_s is the distribution given by Lemma 4.6. Since each time the

coin shows head w.h.p., a union bound shows that it holds that w.h.p. the coin shows head in *all* rounds $r \in B_i$.

COROLLARY 4.7. *If the experiment E_i described above is executed with probability of heads being $1 - 1/D^{c+2}$, where c is the constant fixed in the beginning of the section (in the definition of "approximately equivalent"), w.h.p. no coin flip shows tail. In other words, for each round r, the state of the agent at the end of round $r + \beta \leq \Delta$, $r \in B_i$, is drawn independently from the stationary distribution π.*

4.2.3 Movement on the grid.

Having established that an agent's state can essentially be understood as a (sufficiently small) collection of sets of independent random variables, we focus on the implications on the agents' movement in the grid. Let the random variable $X_r^\uparrow$ have value 1 if the state of the agent at the end of round r is labeled up, and 0 otherwise. Note that these random variables depend only on the state transitions the agent performs in the Markov chain. Also let $X_{\leq r}^\uparrow = \sum_{r'=1}^{r} X_{r'}^\uparrow$.

LEMMA 4.8. *Suppose an agent is initially in a state from the recurrent class C. Then there is a $p^\uparrow \in [0,1]$ only depending on C, such that for each round $r \leq \Delta$ it holds that $\left|X_{\leq r}^\uparrow - rp^\uparrow\right| = o(D/|S|)$ w.h.p.*

PROOF SKETCH. First, we consider the special case of $r = \Delta$. Because w.l.o.g. $\beta = o(D/|S|)$ (follows from the assumptions for the value of χ), it suffices to show that, for a suitable choice of $p^\uparrow$, $\left|\sum_{r'=\beta+1}^{\Delta} X_{r'}^\uparrow - (r-\beta)p^\uparrow\right| = o(D/|S|)$. We will break this sum down into β sums, each considering only the round numbers from the same class B_i.

For each such class B_i, we condition on the event that experiment E_i results in all coin flips showing heads, and so each subsequent state within class B_i is drawn independently from the stationary distribution. Now, we can apply a Chernoff bound to show that w.h.p. the number of moves up in the grid that occur in rounds $r' \in B_i$ does not differ from the expected number of such moves by more than $o(D/(|S|\beta))$. Finally, we show by a union bound that w.h.p. both (1) all coin flips from experiment E_i indeed show heads, and (2) for all i it is true that the number of moves up in the grid that occur in rounds $r' \in B_i$ does not differ from the expected number of such moves by more than $o(D/(|S|\beta))$. The value of $p^\uparrow$ now is simply the sum over i of the expectation for class B_i divided by its size.

For the general case of $r < \Delta$, observe that decreasing r by an integer multiple of β decreases the computed expectation by exactly $p^\uparrow$ (as long as $r > \beta + 1 = o(D/|S|)$). Also, decreasing the number of rounds only decreases the probability of large deviations from the expectation of the random variable. Since $\beta = o(D/|S|)$, the general statement hence follows analogously to the special case. □

Repeating these arguments for the other directions (right, down, and left), we see that overall, each agent behaves fairly predictably. Define $X_{\leq r} \in \mathbb{Z}^2$ to be the random variable describing the sum of all moves the agent performs in the grid up to round r, i.e., its position in the grid (in each dimension) at the end of round r. For this random variable, the following statement holds.

COROLLARY 4.9. *Suppose an agent is initially in a state of the recurrent class C. Then there is $\vec{p} \in [0,1]^2$ depending only on C such that for each $r \leq \Delta$, it holds that $\|X_{\leq r} - r\vec{p}\| = o(D/|S|)$ w.h.p.*

We now resume the proof of Theorem 4.1.

PROOF OF THEOREM 4.1. Denote by $\mathcal{C}$ the set of recurrent classes of the Markov chain representing the state machine of each agent. By Corollary 4.3, it holds for each agent a that, after each round $r \geq R_0 = D^{o(1)}$, w.h.p. the agent is located in recurrent class $C(a) \in \mathcal{C}$. Since Lemma 4.8, and therefore Corollary 4.9, do not depend on the initial state from $C(a)$ the agent is in, the same reasoning shows that, at the end of round r, w.h.p. the position of a will not deviate by more than distance $o(D/|S|)$ from a straight line in the grid. By a union bound, this holds for all agents jointly w.h.p (recall that by assumption n is polynomial in D). Hence, w.h.p., it holds for each agent a and each round $r \geq R_0$ that a never ventures further away from the origin than distance $o(D/|S|)$, or its position does not deviate by more than distance $o(D/|S|)$ from one of at most $|\mathcal{C}|$ straight lines. Since for any straight line only a segment of length $\mathcal{O}(D)$ is in distance $\mathcal{O}(D)$ from the origin, the union of all grid points that are (i) in distance at most D from the origin and (ii) in distance at most $o(D/|S|)$ from one of the straight lines has cardinality $\mathcal{O}(D) \cdot o(D/|S|) \cdot |\mathcal{C}| \leq o(D^2/|S|) \cdot |S| = o(D^2)$. Hence, there is a set $G \subset \mathbb{Z}^2$ of $o(D^2)$ grid points that only depends on the algorithm $\mathcal{A}$, D, and n, with the following property: w.h.p., all grid points in distance D from the origin that are visited within the first Δ steps of an execution of $\mathcal{A}$ are in G. Since there are $\Theta(D^2)$ grid points in distance D from the origin, this implies that the target can be placed in such a way that w.h.p. no agent will find it within $\Delta = D^{2-o(1)}$ rounds, and a uniformly placed target is found in this amount of time with probability $o(1)$. □

Finally, we need to show that Theorem 4.1 also holds with respect to the metric M_{moves}. In the following corollary, we show that each move of an agent on the grid corresponds to at most $D^{o(1)}$ transitions in its Markov chain, or otherwise, the agent does not move on the grid after some point on.

COROLLARY 4.10. *Let $\mathcal{A}$ be an algorithm with $\chi(\mathcal{A}) = b + \log \ell \leq \log\log D - \omega(1)$ and $n \in poly(D)$ agents. There is a placement of the target within distance D from the origin such that w.h.p. none of the n agents executing algorithm $\mathcal{A}$ finds the target during its first (up to) $D^{2-o(1)}$ moves. If the target is instead placed uniformly at random in the square of side length $2D$ centered at the origin, the probability that any of the n agents finds the target within its first (up to) $D^{2-o(1)}$ moves is $o(1)$.*

The proof proceeds by showing that w.h.p., at least one of the following is true about any agent a in any round $r \geq R_0$: (1) a is located in a recurrent class in which all states are labeled "none", or (2) each move on the grid performed by a corresponds to at most $D^{o(1)}$ steps in its Markov chain. In the first case, the agent does not make much progress exploring the grid; in the second case, it follows that $D^{2-o(1)}$ moves in the grid correspond to $D^{2-o(1)}$ transitions in the Markov chain and the result is implied by Theorem 4.1.

5. DISCUSSION AND CONCLUSION

We have presented an algorithm and a lower bound for the problem of n agents searching in a grid for a target placed at distance at most D from the origin. Our lower bound shows that if $\chi(\mathcal{A}) < \log\log D - \omega(1)$ and n is polynomial in D, w.h.p. no agent finds the target within its first $D^{2-o(1)}$

moves. In fact, it is straightforward to generalize the result to $n \in 2^{D^{o(1)}}$, and an exponential number of agents find the target quickly even if performing independent, unbiased random walks. We also present an algorithm $\mathcal{A}$ with $\chi(\mathcal{A}) \leq 3 \log\log D$ that finds the target in $(D + D^2/n)2^{\mathcal{O}(\ell)}$ rounds in expectation, proving our lower bound to be near-tight.

Note that for the upper bound we get stronger results if we consider a fixed search area of radius D centered at the origin, as opposed to aiming at a time complexity that is a function of the actual distance of the target from the origin. In this case, $\chi(\mathcal{A}) = \log\log D + \mathcal{O}(1)$ suffices to find the target in $(D + D^2/n)2^{O(\ell)}$ rounds in expectation.

Concerning open problems, note that there is a gap of $2^{\mathcal{O}(\ell)}$ between the running time of our uniform algorithm and the lower bound. Also, for our lower bound we assume that $\chi(\mathcal{A}) = b + \log\ell < \log\log D - \omega(1)$, but our algorithm requires $b = 3(\log\log D - \log\ell) + O(1)$ bits of memory.

6. REFERENCES

[1] Y. Afek, N. Alon, O. Barad, E. Hornstein, N. Barkai, and Z. Bar-Joseph. A biological solution to a fundamental distributed computing problem. *Science*, 331(6014):183–185, 2011.

[2] S. Albers and M. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164–1188, 2000.

[3] N. Alon, C. Avin, M. Koucky, G. Kozma, Z. Lotker, and M. R. Tuttle. Many random walks are faster than one. In *Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '08, pages 119–128, New York, NY, USA, 2008. ACM.

[4] M. Arbilly, U. Motro, M. W. Feldman, and A. Lotem. Co-evolution of learning complexity and social foraging strategies. *Journal of Theoretical Biology*, 267(4):573 – 581, 2010.

[5] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 269–278, New York, NY, USA, 1998. ACM.

[6] S. Berman, Á. Halász, V. Kumar, and S. Pratt. Algorithms for the analysis and synthesis of a bio-inspired swarm robotic system. In *Swarm Robotics*, pages 56–70. Springer, 2007.

[7] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 355–361 vol. 1, Oct 1990.

[8] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 588–597, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[9] Y. Emek, T. Langner, J. Uitto, and R. Wattenhofer. Ants: Mobile Finite State Machines. In *arXiv.org*, November 2013.

[10] Y. Emek and R. Wattenhofer. Stone Age Distributed Computing. In *ACM Symposium on Principles of Distributed Computing (PODC), Montreal, Quebec, Canada*, July 2013.

[11] O. Feinerman and A. Korman. Memory lower bounds for randomized collaborative search and implications for biology. In *Distributed Computing*, volume 7611 of *Lecture Notes in Computer Science*, pages 61–75. Springer Berlin Heidelberg, 2012.

[12] O. Feinerman and A. Korman. Theoretical distributed computing meets biology: A review. In *ICDCIT*, pages 1–18, 2013.

[13] O. Feinerman, A. Korman, Z. Lotker, and J.-S. Sereni. Collaborative Search on the Plane without Communication. In *Proc. 31st Symposium on Principles of Distributed Computing (PODC)*, pages 77–86, 2012.

[14] W. Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, 2008.

[15] P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48(3):166–177, 2006.

[16] L. Gasieniec, A. Pelc, T. Radzik, and X. Zhang. Tree exploration with logarithmic memory. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 585–594, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[17] L.-A. Giraldeau and T. Caraco. *Social foraging theory*. Princeton University Press, 2000.

[18] R. Harkness and N. Maroudas. Central place foraging by an ant (cataglyphis bicolor fab.): a model of searching. *Animal Behaviour*, 33(3):916 – 928, 1985.

[19] K. Holder and G. Polis. Optimal and central-place foraging theory applied to a desert harvester ant, pogonomyrmex californicus. *Oecologia*, 72(3):440–448, 1987.

[20] C. Lenzen, N. Lynch, C. Newport, and T. Radeva. Trade-offs between selection complexity and performance when searching the plane without communication. In *arXiv.org*, May 2014.

[21] P. Panaite and A. Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33(2):281 – 295, 1999.

[22] S. C. Pratt and D. J. Sumpter. A tunable algorithm for collective decision-making. *Proceedings of the National Academy of Sciences*, 103(43):15906–15910, 2006.

[23] O. Reingold. Undirected st-connectivity in log-space. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 376–385, New York, NY, USA, 2005. ACM.

[24] E. J. Robinson, D. E. Jackson, M. Holcombe, and F. L. Ratnieks. Insect communication: No entry signal in ant foraging. *Nature*, 438(7067):442–442, 2005.

[25] J. S. Rosenthal. Rates of convergence for data augmentation on finite sample spaces. *The Annals of Applied Probability*, pages 819–839, 1993.