Distributed Computing manuscript No.
(will be inserted by the editor)

Searching without Communicating: Tradeoffs Between Performance and Selection Complexity

Christoph Lenzen · Nancy Lynch · Calvin Newport · Tsvetomira Radeva

Abstract We consider the ANTS problem [Feinerman et al.] in which a group of agents collaboratively search for a target in a two-dimensional plane. Because this problem is inspired by the behavior of biological species, we argue that 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 due to selective pressures. Intuitively, the larger the χ value, the more complicated the algorithm, and therefore the less likely it is to arise in nature. In more detail, we propose a new selection complexity metric χ, defined for algorithm $\mathscr{A}$ such that $\chi(\mathscr{A}) = b + \log \ell$, 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 $1/2^{\ell}$). In this paper, we study the trade-off between the standard performance metric of speed-up, which measures how the expected time to find the target improves with n, and our new selection metric. Our goal is to determine the thresholds of algorithmic complexity needed to enable efficient search.

In particular, consider n agents searching for a treasure located within some distance D from the origin (where n is sub-exponential in D). For this problem, we identify the threshold $\log\log D$ to be crucial for our selection complexity metric. We first prove a new upper bound that achieves a near-optimal speed-up for $\chi(\mathscr{A}) \approx \log\log D + \mathscr{O}(1)$. In particular, for $\ell \in \mathscr{O}(1)$, the speed-up is asymptotically optimal. By comparison, the existing results for this problem [Feinerman et al.] that achieve similar speed-up require $\chi(\mathscr{A}) = \Omega(\log D)$. We then show that this threshold is tight by describing a lower bound showing that if $\chi(\mathscr{A}) < \log\log D - \omega(1)$, then with high probability the target is not found in $D^{2-o(1)}$ moves per agent. Hence, there is a sizable gap with respect to the straightforward $\Omega(D^2/n + D)$ lower bound in this setting.

Keywords distributed algorithms · biology-inspired algorithms · search algorithms · Markov chains

Christoph Lenzen
Max Planck Institute for Informatics
E-mail: clenzen@mpi-inf.mpg.de

Nancy Lynch
Massachusetts Institute of Technology
E-mail: lynch@csail.mit.edu

Calvin Newport
Georgetown University
E-mail: cnewport@cs.georgetown.edu

Tsvetomira Radeva (corresponding author)
Massachusetts Institute of Technology
E-mail: radeva@csail.mit.edu

1 Introduction

It is increasingly accepted by biologists and computer scientists that tools of distributed computation can improve our understanding of distributed biological processes [12], [13], [14]. A standard approach is to translate a biological process of interest (ant foraging [12], [14], sensory organ pre-cursor selection [1]) into a formal problem in a distributed computing model, and 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 such an 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 biologically-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 problems inspired from biology 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 a 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.

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 within some (unknown) distance D (measured in number of hops in the grid) from the origin. We assume that n is sub-exponential in D.[1] This problem is described and analyzed in recent work by Feinerman et al. [14], where it is called the *Ants Nearby Treasure Search (ANTS)* problem. The authors in [14] argue that it provides a good approximation of insect foraging, and represents a useful intersection between biological behavior and distributed computation. The analysis in [14] 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 within distance D from the origin.

Selection Metric Motivation. In studying solutions to the ANTS problem, we consider the selection complexity metric χ, which captures the bits of memory and probabilistic range used by a given algorithm. This combined metric is motivated by the fact that memory can be used to simulate small probability values, and small probability values can be used to approximate operations that would otherwise require more memory; for example, given a coin that comes up heads with some small probability $1/2^\ell$ might allow an agent to measure a distance of approximately 2^ℓ without using much memory (e.g., keep moving until the coin comes up heads). In more detail, for algorithm $\mathscr{A}$, we define $\chi(\mathscr{A}) = b + \log \ell$, where b is the number of bits of memory required by the algorithm (note, $b = \log |S|$, where S is the state set of the state machine representation of $\mathscr{A}$), and ℓ is the smallest value such that all probabilities used in $\mathscr{A}$ are bounded from below by $1/2^\ell$. In Sections 3, 4 and 5, 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. 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). On the other hand, using multiple independent events to simulate one with larger bias (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 [14] by now also considering the selection complexity metric χ. We begin by studying lower bounds. We identify $\log\log D$, for a target at distance within D from the origin, as a crucial threshold for the χ metric when studying the achievable speed-up [2] in the foraging problem.

In more detail, our lower bound shows that for any algorithm $\mathscr{A}$ such that $\chi(\mathscr{A}) \leq \log\log D - \omega(1)$, there is a placement of the target within distance D such that the probability that $\mathscr{A}$ finds the target in less than $D^{2-o(1)}$ moves per agent is polynomially small in D, and the probability of finding a target placed randomly within this distance and within $D^{2-o(1)}$ moves is $o(1)$. Since $\Omega(D^2)$ rounds are necessary for a single agent to explore the grid, our lower bound implies that the speed-up of any algorithm for exploring the grid with $\chi \leq \log\log D - \omega(1)$ is bounded from above by $\min\{n, D^{o(1)}\}$. For comparison, recall that because of the trivial $\Omega(D^2/n + D)$ lower bound, the optimal speed-up is $\min\{n, D\}$. At the core of our lower bound is a novel analysis of recurrence behavior of small Markov chains with probabilities of at least $1/2^\ell$.

Concerning upper bounds, we note that the foraging algorithms in [14] achieve near-optimal speed-up in n, but their selection complexity, as measured by $\chi(\mathscr{A})$, is higher than the $\log\log D$ threshold identified by our lower bound: these algorithms require sufficiently fine-grained probabilities and enough memory to randomly generate and store, respectively, coordinates up to distance at least D from the origin; this requires $\chi(\mathscr{A}) \geq \log D$. In this paper, we seek upper bounds that guarantee $\chi(\mathscr{A}) \approx \log\log D$, which is the

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

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

minimum value for which good speed-up is possible. We consider two types of algorithms: non-uniform algorithms in D, which are allowed to use knowledge of D, and uniform algorithms in D, which have no information about D. All our algorithms are non-uniform in n; that is, the algorithms have knowledge of n. We begin by describing and analyzing a simple algorithm that is non-uniform in D and has asymptotically optimal expected running time. The main idea of this algorithm is to walk up to certain points in the plane while counting approximately, and thus using little memory. We can show that this approximate counting 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 $1 + o(1)$.

The main idea of our uniform algorithm is to start with some estimate of D and keep increasing it while executing a corresponding version of our simple search algorithm described above for each such estimate. Similarly to the non-uniform algorithm, the uniform algorithm uses value of χ that is at most $\log\log D + \mathcal{O}(1)$. Additionally, we introduce a mechanism to control the trade-off between the algorithm's running time and number of bits it uses. With that goal, we let the algorithm take as a parameter a non-decreasing function $f(D)$ that represents the running-time overhead we are willing to accept; for a given function $f(D)$, the algorithm guarantees to run in $\mathcal{O}((D^2/n + D) \cdot f(D))$ moves per agent in expectation. We show that the resulting value of the selection metric $\chi = b + \log \ell$ is $\log\log D + \mathcal{O}(1)$, regardless of the choice of f, including $f = \Theta(1)$, in which case the algorithm matches the $\Omega(D^2/n + D)$ lower bound. In Section 4.4, we analyze in detail the resulting χ values for different choices of $f(D)$ and discuss what trade-offs can be achieved between the b and ℓ components of the selection metric. For example, we show that for $f(D) = \Theta(D^{\varepsilon})$, where $0 < \varepsilon < 1$, if ℓ is sufficiently large, then $b = \log\log\log D + \mathcal{O}(1)$ bits are sufficient for the algorithm.

Discussion. From a biological perspective, we do not claim that χ is necessarily the *right* 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 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 [12] and [14], which originally introduced the problem studied here. More precisely, in [14] the authors present an algorithm to find the target in optimal time $\mathcal{O}(D^2/n + D)$ in expectation, assuming that each agent in the algorithm knows the number n of agents (but not D). For unknown n, they show that for every constant $\varepsilon > 0$, there exists a uniform search algorithm that is $\mathcal{O}(\log^{1+\varepsilon} n)$-competitive, but there is no uniform search algorithm that is $\mathcal{O}(\log n)$-competitive. In [12], 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 $\mathcal{O}(\log^{1-\varepsilon} n)$-competitive, the ants need advice size of $\Omega(\log\log n)$ bits. Note that this result also implies a lower bound of $\Omega(\log\log n)$ bits on the total size of the memory of the ants, but only under the condition that close-to-optimal speed-up is required. Our lower bound is stronger in that we show that there is an exponential gap of $D^{1-o(1)}$ for the maximum speed-up (with a sub-exponential number of agents in D). Similarly, the algorithms in [14] need $\Omega(\log D)$ bits of memory, resulting in selection metric value $\chi = \Omega(\log D)$, as contrasted with our algorithm that ensures $\chi = \mathcal{O}(\log\log D)$.

In our previous work in [20], we generalized the ideas of the non-uniform algorithm to show that a uniform algorithm in D can find a target within distance D from the origin in $(D^2/n + D) \cdot 2^{\mathcal{O}(\ell)}$ rounds for $\chi(\mathscr{A}) \leq 3\log\log D + \mathcal{O}(1)$. The running time is asymptotically optimal for $\ell = \mathcal{O}(1)$, but we choose to keep ℓ as a parameter in order to study the trade-off between ℓ and b. While the value of χ needed by the algorithm is asymptotically optimal, it is dominated by the memory component b. In other words, even if the algorithm is given large values of ℓ (corresponding to small probabilities), the algorithm still requires $\Theta(\log\log D)$ bits. One goal for our uniform algorithm in this paper is to allow for a trade-off between the b and ℓ components of the selection metric, potentially at the cost of being some approximation factor over the optimal running time; moreover, our goal is for the algorithm to be also flexible in choosing such an approximation factor.

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] [6] [8], undirected graphs [23] [24], or trees [4] [9]. Out of these, the following papers have restrictions on the memory used in the search: [4] uses $\mathcal{O}(\log n)$ bits to explore an n-node tree, [6] studies the power of a pebble placed on a vertex so that the vertex can later be identified, [9] shows that $\Omega(\log \log n)$ bits of memory are needed to explore some n-node trees, and [24] presents a log-space algorithm for s-t-connectivity. There have been works on graph exploration with multiple agents [3] [10] [16]; while [3] and [16] do not include any memory bounds, [10] presents an optimal algorithm for searching with constant memory and constant-sized messages (in the model introduced in [11]) by agents with very limited computation and communication capabilities. It should be noted that even though these models restrict the agents' memory to very few bits, the fact that the models allow communication makes it possible to simulate larger memory.

So far, in the above papers, we have seen that the metrics typically considered by computer scientists in graph search algorithms are mostly the amount of memory used and the running time. In contrast, biologists look at a much wider range of models and metrics, more closely related to the physical capabilities of the agents. For example, in [5] 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 [25] explores the use of different chemicals used by ants to communicate with one another.

Organization. In Section 2, we present our system model assumptions and formally define the search problem and both the performance and selection metrics that we use to evaluate our algorithms. In Sections 3 and 4, we present our algorithms, starting with a very simple non-uniform algorithm in Section 3 illustrating our main approach. In Section 4, we generalize this approach to algorithms that are uniform in D. In Section 5, we present a lower bound that matches our upper bounds in terms of the selection metric χ. We conclude by discussing some assumptions and possible extensions of our work in Section 6. The appendix contains some definitions and math preliminaries used throughout the technical sections of the paper.

2 Model

Our model is similar to the models in [12] [14]. 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[3]. Without this assumption, any algorithm automatically needs at least $\Omega(\log D)$ bits just to implement the capability to return home. Therefore, while it is a strong assumption, it lets us study the behavior of algorithms with selection complexity $\chi = o(\log D)$. In our setting, the agent returns on a shortest path in the grid that keeps closest to the straight line connecting the origin to its current position. Note that the return path is at most as long as the path of the agent away from the origin; therefore, since return paths increase the running time by at most a factor of two, and we are interested in asymptotic complexity, we ignore the lengths of these paths in our analysis. Next, we give a formal description of our model.

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

For our lower bound in Section 5, it is convenient to use a Markov chain representation of each agent. Therefore, we can express each agent as a Markov chain with transition matrix P, such that for each $s_1, s_2 \in S$, $P[s_1][s_2] = \pi_{s_1}(s_2)$, and start state $s_0 \in S$.

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

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

[3] From a biological perspective, there is evidence that social insects use such a capability by navigating back to the nest based on landmarks in their environment [21].

we ignore the movement of the agent on the way back to the origin, as mentioned earlier in this section.

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

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

Problem Statement. The goal is to find a target located at some vertex at distance (measured in terms of the infinity norm) at most D from the origin in as few expected moves as possible. Note that measuring paths in terms of the max-norm gives us a constant-factor approximation of the actual hop distance. We will consider both non-uniform and uniform algorithms with respect to D; that is, the agents may or may not know the value of D. Technically, in the case of non-uniform algorithms, each different value of D corresponds to a different algorithm. So, we define a family of non-uniform algorithms $\{\mathscr{A}_D\}_{D\in\mathbb{N}}$ where each $\mathscr{A}_D$ is an algorithm with parameter D.

It is easy to see (also shown in [14]) that the expected running time of any algorithm is $\Omega(D+D^2/n)$ even if agents know n and D and they can communicate with each other. This bound can be matched if the agents know a constant-factor approximation of n [14], but as mentioned in Section 1, the value of the selection metric χ (introduced below) in that specific algorithm is $\Omega(\log D)$. For simplicity, throughout this paper we will consider algorithms that are non-uniform in n, i.e., the agents' state machine is allowed to depend on n. One can apply a technique from [14] that the authors use to make their algorithms uniform in n, in order to generalize our results and obtain an algorithm that is uniform in both D and n, at the cost of an $\mathscr{O}(\log^{1+\varepsilon} n)$-factor running time overhead.

Metrics. For the problem defined above, we consider both a performance and a selection metric and study the tradeoffs 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 metric M_{moves}: the minimum over all agents of the number of moves of the agent until it finds the target. Note that for this performance metric we exclude states labeled *none* and *origin* in an execution of an agent. We already argued that the *origin* states increase the running time by at most a factor of two. We consider the transitions to *none* states to be part of an agent's local computation. Intuitively, we can think of consecutive transitions to *none* states to be grouped together with the first transition to a non-*none* state and considered a single move. Both our algorithm bounds and our lower bound are expressed in terms of M_{moves}. For the proof of our lower bound, it is also useful to define a similar metric in terms of the steps of an agent. We define the metric M_{steps} to be the minimum over all agents of the number of steps of the agent until it finds the target. This metric is used only as a helper tool in our lower bound analysis.

The selection metric of a state automaton (and thus of the corresponding algorithm) is $\chi(\mathscr{A}) = b + \log \ell$, where $b = \lceil \log |S| \rceil$ is the number of bits required to encode the states in S and $1/2^\ell$ is a lower bound on $\min\{P[s,s'] \mid s,s' \in S \wedge P[s,s'] \neq 0\}$, that is, on the smallest non-zero probability value used by the algorithm. We further motivate this choice in Section 3 and Section 4, where we describe different trade-offs between the performance metric and the values of b and ℓ.

3 Non-uniform Algorithm

In this section we present an algorithm in which the value of D is available to the algorithm.

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

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

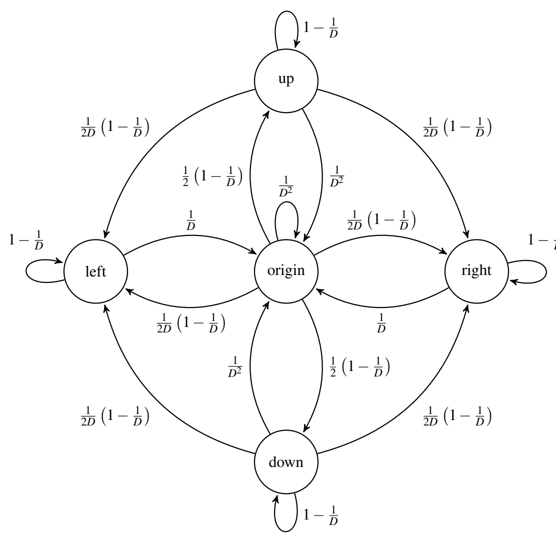

Fig. 1 State machine representation of Algorithm 1. The state names correspond to the values of the labeling function.

Later in this section we present Algorithm $\mathscr{A}_D$, which is a slightly modified version of Algorithm 1 that removes the need for coin $C_{1/D}$. In Theorem 2 we show that Algorithm $\mathscr{A}_D$ guarantees that $\chi = \log\log D + \mathscr{O}(1)$.

Algorithm 1: Non-uniform Search Algorithm.

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

Fix an arbitrary point (x,y) in the grid, where $x,y \in \mathbb{Z}$ and $|x|,|y| \le D$; this point represents the location of the target. The algorithms presented in this section are analyzed with respect to the number of moves until some agent explores grid point (x,y) and, thus, finds the target. For Lemmas 1, 2, 3, and 4 consider an arbitrary fixed agent.

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

Lemma 1 $\mathbb{E}[T] \le 2D$.

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

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

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

Lemma 3 $\mathbb{E}[T \mid U] \le 2\mathbb{E}[T]$.

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

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

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

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

Lemma 4 $E[N] \le 16D$.

Proof First, we bound the probability for the agent to find the target in a single iteration.

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

$$\left(1-\frac{1}{D}\right)^x \left(\frac{1}{D}\right) \ge \left(1-\frac{1}{D}\right)^D \left(\frac{1}{D}\right) \ge \frac{1}{4D}.$$

The probability that the walk right halts after $y \le D$ steps is at least $(1-1/D)^D \ge 1/4$. In each iteration, the probability to find the target is at least $1/(16D)$. The same holds for a target located in any of the other quadrants.

Therefore, $E[N] \le 16D$.

Theorem 1 *Let each of n agents execute Algorithm 1. For a target located within distance D from the origin:*

$$\mathbb{E}[M_{moves}] \le \frac{64D^2}{n} + 6D = \mathscr{O}\left(\frac{D^2}{n} + D\right).$$

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

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

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

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

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

where the second the last step follows by by Lemmas 2, 3, and 4.

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

We now generalize this algorithm to one that uses probabilities lower bounded by $1/2^{\ell}$ for some given $\ell \ge 1$. 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 $\ell \ge 1$.

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

```
for $i = 0 \cdots k$ do
    if $C_{1/2^\ell}$ shows heads then
        return heads
return tails
```

Lemma 5 *Algorithm 2 returns tails with probability* $1/2^{k\ell}$ *and uses* $\lceil \log k \rceil$ *bits of memory.*

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

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

Theorem 2 $\chi(\mathscr{A}_D) = \log\log D + \mathscr{O}(1)$.

Proof By Lemma 5, Algorithm 2 run with parameters $k = \lceil (\log D)/\ell \rceil$ and $\ell' = (\log D)/k \le \ell$, generates coin flips with probability $1/D$ of showing tails. Therefore, the correctness of algorithm $\mathscr{A}_D$ follows from Theorem 1. Since Algorithm 2 does not generate any moves of the agents on the grid, the time complexity of algorithm also follows from Theorem 1. Finally, by Lemma 5 and the fact that Algorithm 1 uses only 3 bits, it follows that

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

4 Uniform Algorithm

In this section, we generalize the results from Section 3 to derive an algorithm that is uniform in D. The main difference is that now each agent maintains an estimate of D that it increases until it finds the target. For each estimate, an agent simply executes a subroutine similar to algorithm $\mathscr{A}_D$. Moreover, the algorithm in this section takes as a parameter a non-decreasing function $f : \mathbb{Z}^+ \to [1,\infty)$ and ensures that the resulting running time $\mathbb{E}[M_{moves}]$[4] is $\mathscr{O}((D^2/n+D)\cdot f(D))$. In other words, given a desired (asymptotic) approximation ratio to the optimal value of $\Theta(D^2/n+D)$, we provide an algorithm that solves the problem in the required expected time and we calculate the necessary value of χ for such a solution. The analysis of the value of $\mathbb{E}[M_{moves}]$ is presented in a general way and works for any function f such that $f(2) \geq 128\ln 8$. For the analysis of the resulting value of the selection metric χ and the trade-off between its components, we plug in different values of f.

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

The rest of this section is organized as follows: Section 4.1 defines a useful sequence of estimates of D using the function f, Sections 4.2 and 4.3 present the algorithm and running time analysis, respectively, and Section 4.4 includes the selection metric analysis for the algorithm.

4.1 Definition and Properties of T_i and D_i

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

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

We define the following set of constraints on the values of the $\mathscr{D}$ and $\mathscr{T}$ sequences. Let $D_0 = 2$.

$$\text{For each } i \in \mathbb{N},\, D_i > 0 \tag{1}$$

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

$$\text{For each } i \in \mathbb{N},\, i \geq 1,\, T_{i+1} = \frac{T_i}{4}\cdot e^{\frac{f(D_{i-1})}{32}\cdot\frac{D_{i-1}^2}{D_i^2}} \tag{3}$$

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

Lemma 6 *Fix n, D_{i-1} and T_i, for any $i \in \mathbb{N}$. Then, Equations (2) and (3) have a unique solution for D_i and T_{i+1}.*

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

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

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

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

Lemma 7 *For each $i \in \mathbb{N}$, $i \geq 1$, $T_{i+1} \geq 2T_i$.*

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

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

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

Case 2: $D_i < 2D_{i-1}$. By Equation (3) and the fact that $f(2) \geq 128 \ln 8$, we have:

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

Note that, based on Lemma 7 and the assumption that f is a non-decreasing function, it follows from Equation (2) that $\mathscr{D}$ is a strictly increasing sequence.

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

$\mathscr{D} : (2, 2.4, 2.9, 3.4, 4.1, 4.9, 5.9, 7.1, 8.4, \cdots)$
$\mathscr{T} : (\perp, 3.2, 4.6, 6.6, 9.4, 13.5, 19.3, 27.6, 39.6, \cdots)$

4.2 Subroutines and Algorithm

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

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

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

Lemma 8 *For a fixed $i \in \mathbb{N}$ and each point (x,y), $x,y \in \mathbb{Z}$ and $|x|,|y| \leq D_i$, if Algorithm 3 is called at the origin, then it visits point (x,y) with probability at least $1/(16D_i)$.*

Proof The proof is identical to the proof of Lemma 4 (D is replaced by D_i).

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

Algorithm 4: Search Algorithm for n agents.

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

4.3 Running Time Analysis of Algorithm 4

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

Throughout the following proofs, we refer to an iteration of the outermost loop as a phase and we refer to a call to search(i) as an iteration. In Lemma 9, we calculate the expected number of moves for an agent to complete phase i. In Lemma 13, we calculate the probability that some agent finds the target in some phase i. Finally, we use these intermediate results to prove the main result of this section, Theorem 3, which shows that the expected number of moves for the first agent to find a target within distance D from the origin is $\mathscr{O}((D + D^2/n) \cdot f(D))$. The structure of the proof is very similar to that of the non-uniform algorithm in Section 3; however, here we consider phases instead of iterations.

For Lemmas 9, 10, 11, 12, and 13 fix some agent and let R_i be the number until the agent completes phase i.

Lemma 9 $\mathbb{E}[R_i] \leq 2T_i$.

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

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

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

Lemma 10 $\mathbb{E}[N_i] \le T_i/D_i$.

Proof By the pseudocode:

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

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

Lemma 11 $\mathbb{E}[R_i \mid U_i] \le 4T_i$.

Proof For each $k \ge 1$, let X_k denote the length of the k'th iteration of phase i. Since the lengths of all iterations in a given phase identically distributed, let $\mathbb{E}[X]$ denote their common expected length. Finally, let U denote the event that a given iteration is unsuccessful. Reasoning identically to Lemma 3, $\mathbb{E}[X \mid U] \le 4D_i$.

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

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

By Lemma 10, we have $\mathbb{E}[R_i \mid U_i] = \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i \mid U_i] \le \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i] \le 4T_i$.

Lemma 12 $\mathbb{E}[R_i \mid S_i] \le 4T_i + 2D$.

Proof For each $k \ge 1$, let X_k denote the length of the k'th iteration of phase i. Since all iterations in a given phase are identical in length, let $\mathbb{E}[X]$ denote their common expected length. Finally, let U denote the event that a given iteration is unsuccessful. Reasoning identically to Lemma 3, $\mathbb{E}[X \mid U] \le 4D_i$. Note that each successful iteration is of length at most $2D$.

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

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

Reasoning similarly to Lemma 11, we get $\mathbb{E}[R_i \mid S_i] \le \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i - 1 \mid S_i] + 2D$. Again, we just need to show that $\mathbb{E}[N_i \mid S_i] \le \mathbb{E}[N_i]$, and the same reasoning as in Lemma 11 holds.

By Lemma 10, we have $\mathbb{E}[R_i \mid S_i] \le \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i \mid S_i] + 2D \le \mathbb{E}[X \mid U] \cdot \mathbb{E}[N_i] + 2D \le 4T_i + 2D$.

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

Lemma 13 *For each phase $i \ge i_0$, the probability that an agent finds the target in phase i is at least* $1 - 2e^{-T_i/(32D_i^2)}$.

Proof By Lemma 10, the expected number of iterations in phase i is at least T_i/D_i.

Fix the number of coin flips performed by the agent in phase i. Since the coin flips are independent, we can apply Chernoff's bound (Inequality (8) in the Appendix), showing that the probability that fewer than $T_i/(2D_i)$ searches are executed in total is at most $e^{-T_i/(12D_i)}$.

Condition on the event that there are at least $T_i/(2D_i)$ iterations in phase i. We can apply Lemma 8 to bound the probability of finding the target in phase i because $D_i \ge D_{i_0} \ge D$, so the probability to miss the target in all iterations of phase i is at most:

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

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

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

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

Theorem 3 *Let each of n agents run Algorithm 4. For a target located within distance D from the origin,* $\mathbb{E}[M_{moves}] = 20(D^2/n + 2D) \cdot f(D) = \mathcal{O}(D^2/n + D) \cdot f(D)$.

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

By Lemma 13, the probability that none of the agents find the target in phase i is at most

$$2e^{-nT_i/(32D_i^2)} \leq 2e^{-(f(D_{i-1})/32)(D_{i-1}^2/D_i^2)} = p_i. \tag{4}$$

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

$$\Pr[N = j] \leq \prod_{i=i_0}^{i_0+j-1} p_i. \tag{5}$$

Next, note that the value of p_i appears in the definition of T_i in Equation (3). So, we can write $T_{i+1} = T_i/(2p_i)$. Also, for any $j \geq 0$:

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

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

$$\sum_{i=j}^{k} T_i \leq T_k \sum_{i=j}^{k} 2^{j-k} \leq 2T_k. \tag{6}$$

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

$$\begin{aligned}
\mathbb{E}[M_{moves}] &\leq \sum_{i=1}^{i_0-1} \mathbb{E}[R_i] + \mathbb{E}\left[\sum_{i=i_0}^{i_0+N-1} R_i\right] + \mathbb{E}[R_{i_0+N}] \\
&\leq \sum_{i=1}^{i_0-1} \mathbb{E}[R_i] + \sum_{j=0}^{\infty} \sum_{i=i_0}^{i_0+j-1} \mathbb{E}[R_i \mid N = j] \cdot \Pr[N = j] \\
&\quad + \sum_{j=0}^{\infty} \mathbb{E}[R_{i_0+j} \mid N = j] \cdot \Pr[N = j] \\
&\leq \sum_{i=1}^{i_0-1} \mathbb{E}[R_i] + \sum_{j=0}^{\infty} \sum_{i=i_0}^{i_0+j-1} \mathbb{E}[R_i \mid U_i] \cdot \Pr[N = j] \\
&\quad + \sum_{j=0}^{\infty} \mathbb{E}[R_{i_0+j} \mid S_i] \cdot \Pr[N = j].
\end{aligned}$$

Recall that by Lemmas 9, 11, and 12, we have $\mathbb{E}[R_i] \leq 2T_i$, $\mathbb{E}[R_i \mid U_i] \leq 4T_i$, and $\mathbb{E}[R_i \mid S_i] \leq 4T_i + 2D$, respectively.

$$\begin{aligned}
\mathbb{E}[M_{moves}] &\leq \sum_{i=1}^{i_0-1} 2T_i + \sum_{j=0}^{\infty} \sum_{i=i_0}^{i_0+j-1} 4T_i \cdot \Pr[N = j] \\
&\quad + \sum_{j=0}^{\infty} (4T_{i_0+j} + 2D) \cdot \Pr[N = j] \\
&\leq \sum_{i=1}^{i_0-1} 2T_i + \sum_{j=0}^{\infty} \sum_{i=i_0}^{i_0+j} 4T_i \cdot \Pr[N = j] + 2D.
\end{aligned}$$

Then, by Equations (4), (5), and (6), it follows that:

$$\begin{aligned}
\mathbb{E}[M_{moves}] &\leq 4T_{i_0} + \sum_{j=0}^{\infty} 8T_{i_0+j} \cdot \Pr[N = j] + 2D \\
&\leq 4T_{i_0} + 8T_{i_0} \sum_{j=0}^{\infty} \prod_{i=i_0}^{i_0+j-1} \left(\frac{1}{2p_i}\right) \prod_{i=i_0}^{i_0+j-1} p_i + 2D \\
&\leq 4T_{i_0} + 8T_{i_0} \sum_{j=0}^{\infty} 2^{-j} + 2D \\
&\leq 4T_{i_0} + 16T_{i_0} + 2D.
\end{aligned}$$

Finally, by Equation (2) and $D_{i_0-1} \leq D \leq D_{i_0}$:

$$\begin{aligned}
\mathbb{E}[M_{moves}] &\leq 20\left(\frac{D_{i_0-1}^2}{n}\right) f(D_{i_0-1}) + 2D \\
&\leq 20\left(\frac{D^2}{n} + 2D\right) f(D).
\end{aligned}$$

As a technical note, if the right hand side of Equation (5) does not go to 0 as j goes to infinity, then we cannot use the method above to bound $\mathbb{E}[M_{moves}]$ because $\Pr[N = \infty] \neq 0$ implying that we may need an unbounded number of phases to find the target. However, this is not the case because it is easy to see that each phase terminates in a finite and bounded number of rounds with probability 1, so $\Pr[N = \infty] = 0$.

4.4 Selection Metric Analysis

In this section, we analyze Algorithm 4 with respect to the selection metric. In Section 4.4.1, we prove some bounds on the χ selection metric for an arbitrary function f (subject to the constraints listed at the beginning of Section 4) used to define the sequences $\mathcal{D}$ and $\mathcal{T}$. Next, in Sections 4.4.2 and 4.4.3, we substitute some specific functions for f in order to get closed-form results for some different values of χ. For the memory component, b, of the selection metric, we consider only dynamically-changing memory (variables that take on different values throughout the execution of the algorithm); for example, the loop counter i and the corresponding value T_i/D_i in Algorithm 4 are dynamically changing, while the entire pre-computed sequences of T_i's and D_i's are not dynamically changing because the algorithm uses them only as a look-up table and does not modify these sequences.

4.4.1 General Analysis

The memory requirements of the algorithm can be split into three parts: (1) bits to represent the counter value i, (2) bits to implement the search routine with argument i, and (3) bits to implement coin $C_{1/x}$ for $x = T_i/D_i$. Similarly to Section 4.3, let i_0 be the minimum phase such that $D \le D_{i_0}$.

Note that, technically, these memory requirements are random variables because the algorithm does not guarantee that once it reaches phase i_0 it finds the target with probability 1. In other words, it is possible for the algorithm to reach phases greater than i_0 before actually finding the target. Therefore, for the purposes of this analysis, we will assume that after the algorithm reaches phase i_0 it may run out of memory, and if it does so, it just stops incrementing the phases. Since it has already reached the right distance to search, by the analysis in Section 4.3, we know that this restriction does not prevent the algorithm from finding the target.

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

For part (2), we can use the subroutine in Algorithm 2 in order to implement the specific coin values we need using only the coin we have, $C_{1/2^\ell}$ (recall that ℓ is a parameter that determines the smallest probability the algorithm may use). By Lemma 5, it follows that we need at most $\log\log D_{i_0} - \log\ell$ bits to implement coin $C_{1/D_{i_0}}$. The remaining part of the search subroutine uses only a constant number of bits.

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

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

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

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

4.4.2 $f(D_i) = c$ for some constant $c \ge 128$

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

Proof Substituting f in Equations (2) and (3), we get the following equations:

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

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

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

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

$$\frac{D_i}{D_{i-1}} \ge \sqrt{\frac{3}{2\ln 4}} > 1.$$

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

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

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

Note that, for this choice of f, the value of b used by the algorithm is $\Theta(\log\log D)$ in the worst case, regardless of the value of ℓ. This is true because larger values of ℓ can help implement the various coins used in the algorithm with fewer bits, but it does not affect the fact that $\Theta(\log\log D)$ bits are used to represent index i_0. Therefore, even if large values of ℓ are available to the algorithm, the memory requirement remains the same.

Next, we generalize Theorem 4 to any function f that is polynomial in D.

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

Proof First, by Theorem 4, we know that Algorithm 4 uses $\chi = 2\log\log D + \mathcal{O}(1)$, for $f(D_i) = c$, where $c \geq 128$. A faster growing function results in faster growing values of the subsequent D_i's, so the value D_{i_0} is reached faster.

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

4.4.3 $f(D_i) = \Theta(D^\varepsilon)$ *for some* $0 < \varepsilon < 1$.

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

Theorem 5 *For* $f(D_i) = \Theta(D_i^\varepsilon)$*, where* $0 < \varepsilon < 1$*, and* $\ell = \log D - \log\log D$*, Algorithm 4 uses* $b = \log\log\log D + \mathcal{O}(1)$.

Proof Substituting f in Equations (2) and (3), we get the following equations:

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

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

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

Given that $D_0 = 2$, it is easy to see that each D_i is at least $\Theta(2^{(1+\Omega(\varepsilon))^{i-1}})$. Therefore,

$$i_0 - 2 = \mathcal{O}(\log\log D_{i_0-1}) = \mathcal{O}(\log\log D),$$

so the algorithm uses at most $\log\log\log D + \mathcal{O}(1)$ bits to represent index i_0.

4.4.4 Discussion

First, note that some other functions, not considered above, lie asymptotically between $\Theta(1)$ and $\Theta(D_i^\varepsilon)$. For example, two such functions are $\Theta(\log D_i)$ and $2^{\Theta(\log^\lambda D_i)}$ where $\lambda \in [0,1]$. We can perform similar calculations to those in Sections 4.4.2 and 4.4.3, to show that, for both of these functions, the algorithm uses $\mathcal{O}(\log\log D)$ bits to encode the index i_0. In other words, we get the same asymptotic bounds for χ as in the case of $f(D_i) = \Theta(1)$. In contrast, $f(D_i) = \Theta(D^\varepsilon)$ is the slowest-growing function we could identify for which (given a large enough value of ℓ) the memory used by the algorithm is $\mathcal{O}(\log\log\log D)$ bits, substantially smaller that $\mathcal{O}(\log\log D)$. This implies that as the desired approximation to the running time, specified by f, grows to $\Theta(D^\varepsilon)$ or higher (e.g. polynomial in D), the memory used by the algorithm decreases to $\mathcal{O}(\log\log\log D)$ bits. Functions larger that polynomial in D are not of particular interest here because for the resulting running time there are much simpler ways to search the plane, for example, simple random walks.

The selection metric bounds in this section can be shown to be generalizations of the one we get in [20] where the approximation factor of the running time is $2^{\mathcal{O}(\ell)}$. Performing similar calculations to those in Section 4.4.2, we can see that in the case of $f(D_i) = 2^{\mathcal{O}(\ell)}$, the estimates of D grow as $D_i/D_{i-1} \geq 2^{\mathcal{O}(\ell)}$. Therefore, we need a selection metric value of $\chi = \mathcal{O}(\log\log D)$ for the algorithm to satisfy this approximation factor, which is asymptotically the same as in [20].

5 Lower bound

In this section, we present a lower bound on the number of rounds necessary for an algorithm to find a target placed within distance D from the origin, with non-negligible probability, if the algorithm satisfies $\chi(\mathscr{A}) \leq \log\log D - \omega(1)$. The rest of this section is structured as follows: in Section 5.1, we state the main theorem with respect to the metric M_{steps} and non-uniform algorithms and give an overview of the proof, in Section 5.2, we present the proof in detail, and, finally, in Sections 5.3 and 5.4, we extend the lower bound to the case of the metric M_{moves} for non-uniform and uniform algorithms, respectively.

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

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

Theorem 6 *For each $D \in \mathbb{N}$, such that $D > 1$, and for each $n \in \mathbb{N}$, $n \le T(D)$, assume algorithm $\mathscr{A}_D$ with n agents satisfies $\chi(\mathscr{A}_D) = b + \log\ell \le \log\log D - f_1(D)$. Then, there exists a placement (x,y), $|x|,|y| \le D$ of the target, such that, with probability at least $1 - 1/D^c$, algorithm $\mathscr{A}_D$ satisfies $M_{steps} > D^{2-f_2(D)}$ for this placement (x,y).*

Proof Overview: Here we provide a high-level overview of our main proof argument. We fix an algorithm $\mathscr{A}_D$ for $D \in \mathbb{N}$, $D > 1$, and focus on executions of this algorithm of length $D^{2-o(1)}$ rounds. We prove that since agents have $o(\log D)$ states, they "forget" about past events too fast to behave substantially differently from a biased random walk. Note that a random walk is essentially memoryless since each new step is independent of the previous steps, so it cannot "remember" what it has visited already.

More concretely, first we show in Corollary 2 that after $D^{o(1)}$ initial rounds each agent is located in some recurrent class C of the Markov chain. We use this corollary to prove, in Corollary 3, that after the initial $D^{o(1)}$ rounds each agent either does not return to the origin, or it keeps returning every $D^{o(1)}$ rounds, so it does not explore much of the grid. Therefore, throughout the rest of the proof we can ignore the states labeled "origin".

Assume (for the purposes of this overview) that there is a unique stationary distribution of C[5]. Since there are few states and non-zero transition probabilities are bounded from below, standard results on Markov chains imply that taking $D^{o(1)}$ steps from any state in the recurrent class will result in a distribution on the states of the class that is (almost) indistinguishable from the stationary distribution; in other words, any information agents try to preserve in their state will be lost quickly with respect to D.

The next step in the proof is a coupling argument. We split up the rounds in the execution into groups such that within each group, rounds are sufficiently far apart from one another for the above "forgetting" to take place. For each group, we show that drawing states independently from the stationary distribution introduces only a negligible error (Lemma 16 and Corollary 5). So, we can apply a Chernoff bound to each group, yielding that an agent will not deviate substantially from the expected path it takes when, in each round, it draws a state according to the stationary distribution and executes the corresponding move on the grid (Lemma 18 and Corollary 7). Taking a union bound over all groups, it follows that, with high probability, each agent will not deviate from a straight line (the expected path associated with the recurrent class it ends up in) by more than distance $o(D/|S|)$, where S is the number of states of the Markov chain. It is crucial here that the corresponding region in the grid, restricted to distance D from the origin, has size $o(D^2/|S|)$ and depends only on the component of the Markov chain the agent ends up in. Therefore, since there are no more than $|S|$ components, taking a union bound over all agents shows that with high probability together they visit an area of $o(D^2)$.

5.2 Proof

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

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

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

Consider the distribution of executions of $\mathscr{A}_D$ of length $R_0 + \Delta$ rounds. We break the proof down into three main parts. Sections 5.2.1 and 5.2.2 use standard Markov chain techniques to derive some results for our constrained (in terms of number of states and range of probabilities) Markov chain, and Section 5.2.3 applies these results to the movement of the agents in the grid. First, in Section 5.2.1, we show that, with high probability, after a certain number of initial rounds each agent is in a recurrent class of its Markov chain. Until we resume the proof of Theorem 6, we also condition on this recurrent class not containing any states

[5] This holds only if the induced Markov chain on the recurrent class is aperiodic, but the reasoning is essentially the same for the general case. We handle this technicality at the beginning of Section 5.2.2.

labeled *origin*. Next, in Section 5.2.2, we show that if we break down the execution into sufficiently large blocks of rounds, then we can assume that, with high probability, the steps associated with rounds in different blocks do not depend on each other. Finally, in Section 5.2.3, we focus on the movement of the agents in the grid, derived from these "almost" independent steps, and we show that with high probability, among all points at distance $\mathcal{O}(D)$ from the origin, the agents will only explore a total area of $o(D^2)$.

5.2.1 Initial steps in the Markov chain

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

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

Lemma 14 *Let s be an arbitrary state. Then, with probability at least* $1 - 1/D^{c'}$*, one of the following is true: (1) the agent visits state s within* R_0 *rounds, or (2) the agent is located in some state* s' *immediately after* $r \le R_0$ *rounds such that s is not reachable from* s'.

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

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

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

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

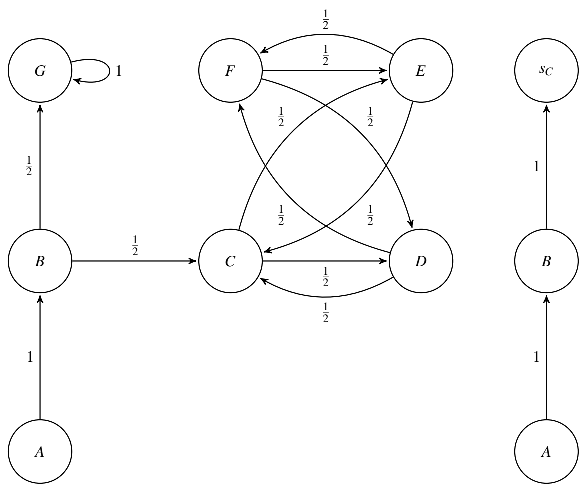

Fig. 2 On the left: simple example of Markov chain with start state A. The recurrent classes are $\{G\}$ and $\{C,D,E,F\}$. On the right: all recurrent states merged into a single state s_C.

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

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

Corollary 2 *With probability at least* $1 - 1/D^{c'}$*, it is true that* $C(R_0) \neq \perp$.

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

By definition of a recurrent class and because there is only one such class, for each state s in the derived Markov chain, the recurrent state s_C is always reachable from s. By Lemma 14, with probability at least $1-1/D^{c'}$, the agent visits s_C within R_0 rounds. This implies that in the original Markov chain, with probability at least $1-1/D^{c'}$, the agent visits *some* recurrent state $s \in C(R_0)$, such that $C(R_0) \neq \perp$, within R_0 rounds.

In Corollary 2, we showed that, with high probability, within R_0 rounds the agent is located in some recurrent class $C(R_0) \neq \perp$. Since the agent does not leave that class in subsequent rounds, we will refer to it by C (a random variable). Finally, we show that, with high probability, either recurrent class C does not contain any states labeled *origin*, or the agent keeps returning to the origin often.

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

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

Otherwise, each state $s' \in C$ labeled *origin* is reachable from state s in each round $r \ge R_0$. By Lemma 14, with probability at least $1-1/D^{c'}$, the agent visits state s' within R_0 rounds. Since the agent does not leave C, we can repeat this argument for each group of R_0 rounds in the execution. In an execution of length $R_0+\Delta$ rounds, there are $o(D^2)$ groups of R_0 rounds. By a union bound, with probability at least $1-1/D^{c'-2}$, for all rounds r, where $R_0 \le r \le \Delta + R_0$, the agent visits a state labeled origin at least once between rounds r and $r+R_0$.

By the law of total probability, since all execution prefixes of R_0 rounds are disjoint, the conclusion above holds for all executions. Combining this result and Corollary 2 by a union bound shows that, with probability at least $1-1/D^{c'-3}$, at least one of the two statements of the corollary holds.

Until we resume the proof of Theorem 6, we consider executions after round R_0 and condition on the event that the agent is in some recurrent class C which does not contain any states labeled *origin*. In the proof of Theorem 6 at the end of this section, we refer to Corollary 3 in order to incorporate the probability of this event into the final probability bound. For convenience, we refer to the remaining Δ rounds of the execution as round numbers 1 to Δ. This numbering is used throughout Sections 5.2.2 and 5.2.3; at the end of Section 5, when we resume the proof of Theorem 6, we incorporate the initial rounds to conclude the final result about the entire execution.

5.2.2 Moves drawn from the stationary distribution

Fix an arbitrary recurrent class C of the Markov chain. Let t denote the period of the Markov chain (an aperiodic chain has period $t=1$). We apply Theorem 9 in the Appendix to C and denote by $G_1,\cdots,G_t$ the equivalence classes based on the period t whose existence is guaranteed by the theorem.

Consider blocks of rounds, each of which consists of $\beta = 2d|S|^2 p_0^{-2|S|^2} \ln D = D^{o(1)}$ rounds. We assume that β is a multiple of t. Otherwise, we can use $t\lceil \beta/t \rceil = \mathscr{O}(\beta)$ assuming $t \in \mathscr{O}(\beta)$; this is true because $t \le |S|$ and $p_0^{-2|S|^2} \ln D \ge 1$ because of the restriction on χ. We define groups of rounds such that each group contains one round from each block. Formally, for $1 \le i \le \beta$ and $j \in \mathbb{N}_0$, group B_i contains round

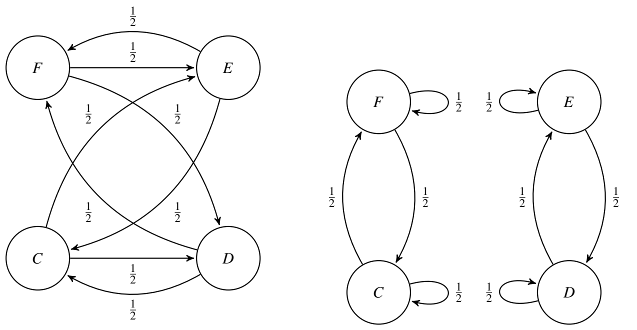

Fig. 3 On the left: a recurrent class, with transition matrix P and period 2, of the simple example of Markov chain from Figure 2. On the right: Markov chain induced by P^2. The equivalence classes here are $\{C,F\}$ and $\{D,E\}$.

numbers $i + j\beta \le \Delta$. Observe that, based on this definition, immediately after each round from a given group, the agent is in some state from the same class $G \subseteq C$ that is recurrent and closed under P^t, where P is the probability matrix of the original Markov chain. By [15][Chapter XV.7], there is a unique stationary distribution π of the Markov chain on G induced by P^t (see Figure 3 for an illustration of P^t and the equivalence classes $G_1,\cdots,G_t$).

The following lemma bounds the value of $\pi(s')$ for each state $s' \in G$ in the Markov chain on G induced by P^t.

Lemma 15 *Assume $|G|>1$. Then, for each $s' \in G$, and each constant $c'' \ge 2^{-f_1(D)}$:*

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

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

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

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

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

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

We say that two discrete probability distributions π_1 and π_2, with the same domain, are *D-approximately equivalent* iff $\|\pi_1 - \pi_2\| \le 1/D^d$ where $\|\cdot\|$ denotes the ∞-norm on the given space.

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

Next, we show that the distribution π_s of the Markov chain is D-approximately equivalent to the stationary distribution π of the Markov chain. We obtain the following corollary of Lemma 2 from [26] (also stated as Lemma 19 in the Appendix) applied to the Markov chain induced by the matrix P^t restricted to class G.

Corollary 4 *For each $s \in G$, π_s and π are D-approximately equivalent.*

Proof We can apply Lemma 19 in order to show that the stationary distribution π and the actual distribution π_s are very close to each other (D-approximately equivalent).

Since β is a multiple of t, we can consider the probability matrix P^t, which by Theorem 9 induces a Markov chain on G. Also, since the Markov chain induced by P^{tk_0} is aperiodic and since C is a recurrent class, we can apply Lemma 20. It essentially states that there exists an integer r such that there is a walk of length exactly r between any pair of states in the Markov chain. Moreover, we are guaranteed that $r \le 2|S|^2$.

Next, we apply Lemma 19 to this chain with the following parameters: $k_0 = r/t$, $Q(s) = 1$ (i.e., $Q(s') = 0$ for all $s' \in G \setminus \{s\}$), and $k = \beta/t$. We also need that $P^{tk_0}(s', s) \ge \varepsilon$ for each $s' \in G$ and a suitable $\varepsilon > 0$. We can choose $\varepsilon = p_0^{2|S|^2}$ guaranteeing that $P^{tk_0}(s', s) = P^r(s', s) \ge p_0^r \ge p_0^{2|S|^2} = \varepsilon$.

By Lemma 19, it follows that:

$$\begin{aligned}\|\pi_s - \pi\| &\le (1-\varepsilon)^{\lfloor k/k_0 \rfloor} = \left(1 - p_0^{2|S|^2}\right)^{dp_0^{-2|S|^2} \ln D} \\ &\le e^{-d \ln D} = \frac{1}{D^d}.\end{aligned}$$

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

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

Lemma 16 *Let $1 \le i \le \beta$ and $\tau = i \bmod t$ for some integer τ. Then, for each state $s \in G$ there exists a probability distribution π'_s such that:*

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

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

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

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

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

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

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

By Corollary 4, π_s and π are D-approximately equivalent, the bound on $\pi(s')$ from Lemma 15, and the assumptions $d > 2(c'+1)$ and $c'' = 2^{-f_1(D)} < 1$. Similarly,

$$\begin{aligned}\pi'_s(s') &\ge \frac{\pi(s')}{D^{c'+2}} - D^{c'+2}\|\pi_s - \pi\| \\ &\ge \frac{1}{D^{c''}D^{c'+2}} - \frac{1}{D^{d-c'-2}} \ge 0,\end{aligned}$$

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

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

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

In the original Markov chain, for each round $r \in B_i$, immediately after which the agent is in state s, the state immediately after round $r + \beta$ is determined based on the distribution π_s. In the modified Markov chain M, the state immediately after round $r + \beta$ is determined by π from state A_s,

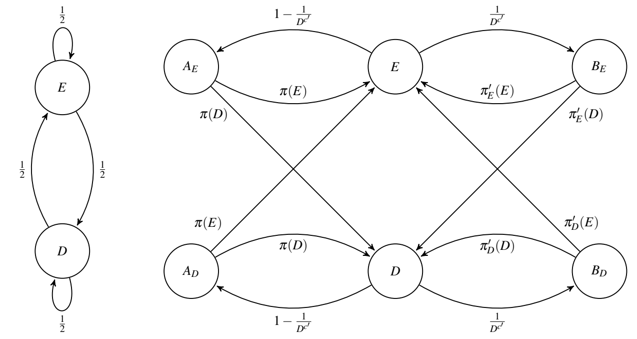

Fig. 4 On the left: equivalence class $\{E,D\}$ induced by P^2 from Figure 3. On the right: derived Markov chain M, ignoring the exact probabilities on the left.

and by π'_s from state B_s. By Lemma 16, it is clear that the distribution of states visited in rounds $r \in B_i$ in the original Markov chain is the same as the distribution in the corresponding rounds of the modified Markov chain.

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

Corollary 5 *For each i, $1 \leq i \leq \beta$, $P[\mathscr{E}_i] \geq 1 - 1/D^{c'}$.*

Proof Consider the coin flips in all rounds $r \in B_i$ in which the Markov chain M is in some state s; these coin flips determine whether the next state is A_s or B_s. By the definition of the modified Markov chain M, with probability at least $1 - 1/D^{c'}$, the next state is A_s. By a union bound, the probability that the next state is A_s for all rounds $r \in B_i$ (whose number is Δ/β where $\beta = D^{o(1)} < D^2$) is:

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

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

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

5.2.3 Movement on the grid

Next, we focus on the implications of the results in the previous sections on the agents' movement in the grid. In order to use Corollary 5, we will base the results of this subsection on the behavior of the derived Markov chain M. However, since we are only reasoning about rounds from blocks B_i for some i, as we already mentioned, by Lemma 16, the distribution of states in the derived Markov chain M is the same as in the original Markov chain. Therefore, the results about the movement of the agents on the grid based on Markov chain M also apply to the movement of the agents on the grid in the original Markov chain.

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

Recall that $\mathscr{E}_i$ denotes the event that for all rounds $r \in B_i$, the state in Markov chain M immediately after round $r+\beta$ is drawn from the stationary distribution. By Corollary 5, $\mathscr{E}_i$ occurs with probability at least $1 - 1/D^{c'}$.

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

Lemma 17 *For each i, where $1 \leq i \leq \beta$, and each round $r \leq \Delta$, conditioning on event $\mathscr{E}_i$, with probability at least $1 - 1/D^{c'-1}$, it is true that:*

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

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

By linearity of expectation,

$$\begin{aligned}\mu_i = \mathbb{E}\left[\sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} X_{r'}^{\uparrow} \,\middle|\, \mathscr{E}_i \right] &= \sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} \mathbb{E}\left[X_{r'}^{\uparrow} \,\middle|\, \mathscr{E}_i \right] \\ &= \sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} p_i^{\uparrow} = p_i^{\uparrow} \left\lfloor \frac{r}{\beta} - 1 \right\rfloor.\end{aligned}$$

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

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

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

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

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

$$Z_r^{\uparrow} = \sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} X_{r'}^{\uparrow} \Big| \mathscr{E}_i + \sum_{y=1}^{\lceil 3c' \ln D - \mu_i \rceil} Y_y^{\uparrow}.$$

By linearity of expectation:

$$\begin{aligned} E[Z_r^{\uparrow}] &= \mu_i + \lceil 3c' \ln D - \mu_i \rceil \cdot \frac{\max\{0, 3c' \ln D - \mu_i\}}{\lceil 3c' \ln D - \mu_i \rceil} \\ &= \max\{\mu_i, 3c' \ln D\}. \end{aligned}$$

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

By a Chernoff bound with $\delta = \sqrt{3c' \ln D / E[Z_r^{\uparrow}]}$, it follows that:

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

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

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

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

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

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

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

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

Lemma 18 *There exists $p^{\uparrow} \in [0,1]$, such that for each round $r \leq \Delta$, with probability at least $1 - 1/D^{c'-2}$, it holds that $\left|X_{\leq r}^{\uparrow} - rp^{\uparrow}\right| = o(D/|S|)$.*

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

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

By Lemma 17, we know that for each i, conditioned on $\mathscr{E}_i$, with probability at least $1 - 1/D^{c'-1}$ it holds that:

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

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

$$\begin{aligned} \mathbb{E}\left[X_{\leq r}^{\uparrow} \middle| \bigwedge_{i=1}^{\beta} \mathscr{E}_i\right] &\leq \sum_{i=1}^{\beta} \mathbb{E}\left[\sum_{\substack{\beta+1 \leq r' \leq r \\ r' \in B_i}} X_{r'}^{\uparrow} \middle| \mathscr{E}_i\right] + \beta \\ &= r \sum_{i=1}^{\beta} \frac{p_i^{\uparrow}}{\beta} + \beta. \end{aligned}$$

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

By a union bound, $\bigwedge_i \mathscr{E}_i$ occurs with probability at least $1 - 1/D^{c'-1}$ because there are $\beta = o(D)$ such events and each one of them holds with probability at least $1 - 1/D^{c'}$, by Corollary 5. By another union bound, with probability at least $1 - 1/D^{c'-2}$, both $\bigwedge_i \mathscr{E}_i$ occurs and Lemma 17 holds for all i. By the definition of β, it follows that $\beta = o(D/|S|)$. Also, since, the expected number of moves *up* in β rounds is at most β, and the actual number of such moves differs by at most β from the expectation, it follows that:

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

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

Corollary 6 *1. There exists $p^{\rightarrow} \in [0,1]$, such that for each round $r \le \Delta$, with probability at least $1-1/D^{c'-2}$, it holds that $\left|X^{\rightarrow}_{\le r} - rp^{\rightarrow}\right| = o(D/|S|)$.*

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

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

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

Corollary 7 *There exists $\mathbf{p} \in [-1,1]^2$, such that for each $r \le \Delta$, with probability at least $1-1/D^{c'-3}$, $\|X_{\le r} - r\mathbf{p}\| = o(D/|S|)$.*

Proof Observe that $X_{\le r} = (X^{\uparrow}_{\le r} - X^{\downarrow}_{\le r}, X^{\rightarrow}_{\le r} - X^{\leftarrow}_{\le r})$. Hence, setting $\mathbf{p} = (p^{\uparrow} - p^{\downarrow}, p^{\rightarrow} - p^{\leftarrow})$, by Lemma 18, Corollary 6 and a union bound, it follows that $\|X_{\le r} - r\mathbf{p}\| = o(D/|S|)$ with probability at least $1-1/D^{c'-3}$.

We are now ready to resume the proof of Theorem 6.

Proof (Proof of Theorem 6) Denote by $\mathscr{C}$ the set of recurrent classes of the original Markov chain of each agent. By Corollary 2, it holds for each agent that, with probability at least $1-1/D^{c'-3}$, the agent is located in some recurrent class $C(a) \in \mathscr{C}$ within R_0 rounds. By Corollary 3, with probability at least $1-1/D^{c'-3}$, either the agent visits a state labeled origin every R_0 rounds, or none of the states in C are labeled *origin*. In the first case, then the agent does not visit a point in the grid at distance more than $R_0 = D^{o(1)}$ from the origin. In the second case, we can apply Corollary 7 to conclude that, with probability at least $1-1/D^{c'-3}$, the position of the agent does not deviate by more than distance $o(D/|S|)$ from a straight line in the grid starting at the origin and ending at point $\Delta\mathbf{p}$ ($\mathbf{p}$ depends only on $C(a)$). Therefore, by a union bound with the results from Corollary 2, it follows that with probability at least $1-1/D^{c'-4}$, either an agent does not venture further away from the origin than distance $o(D/|S|)$, or its position does not deviate by more than distance $o(D/|S|)$ from one of at most $|\mathscr{C}|$ straight lines or the origin. By a union bound, this holds for all agents jointly with probability at least $1-1/D^{c'-4-c_n} = 1/D^c$ (recall that by assumption $n \le T(D) \le D^{c_n}$).

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

In the above proof, note that if the target is placed uniformly at random in the square with side length $2D$ centered at the origin, then it is no longer true that no algorithm finds it in the specified amount of time. In fact, any algorithm that explores at least one grid point has a $\Omega(1/D^2)$ probability of finding a uniformly-placed target. Thus, the correctness of Theorem 6 relies on the fact that the target is placed in the grid adversarially.

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

First, we show that Theorem 6 also holds with respect to the metric M_{moves}. In the following corollary, we show that either each move of an agent on the grid corresponds to at most $D^{o(1)}$ transitions in its Markov chain, or the agent does not move on the grid after some point on. Therefore, since Theorem 6 guarantees that, with high probability, no agent finds the target in $D^{2-o(1)}$ *steps*, then it must be true that, with high probability, no agent finds the target in $D^{2-o(1)}$ *moves*.

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

Corollary 8 *For each $D \in \mathbb{N}$, $D > 1$ and $n \in \mathbb{N}$, $n \le T(D)$, suppose algorithm $\mathscr{A}_D$ with n agents satisfies $\chi(\mathscr{A}_D) = b + \log\ell \le \log\log D - f_1(D)$. There exists a placement (x,y), $|x|,|y| \le D$ of the target, such that, with probability at least $1-1/D^c$, algorithm $\mathscr{A}_D$ satisfies $M_{moves} > D^{2-f_3(D)}$ for this placement (x,y).*

Proof The setup for this proof is the same as that for Theorem 6, so we use the same constants and values defined in Section 5.2. The results from Section 5.2.1 hold with respect to these constants and values, so we can reuse them here.

Consider any fixed execution prefix of length R_0 rounds in which an agent is in some state s in some recurrent class C. By Corollary 2, this is true with probability at least $1-1/D^{c'}$. If C contains only states labeled *none*, then the agent does not make any progress in the grid after it reaches its recurrent class, so it does not visit more than R_0 grid points.

Otherwise, if C contains a state s', labeled *up*, *down*, *left*, or *right*, we show that it is reachable from state s after

r rounds such that $R_0 \leq r \leq 2R_0$. By Lemma 14, with probability at least $1 - 1/D^{c'}$, the agent visits state s' within R_0 rounds. In an execution of length $R_0 + \Delta$, there are $o(D^2)$ groups of R_0 rounds. By a union bound, with probability at least $1 - 1/D^{c'-2}$, the agent visits a state labeled *up*, *down*, *left*, or *right* at least Δ/R_0 times. By the law of total probability, since all execution prefixes of length R_0 are disjoint, this conclusion holds for all executions. By a union bound, this result and Corollary 2 hold jointly with probability at least $1 - 1/D^{c'-3}$.

By Theorem 6, there is a placement of the target such that, with probability at least $1 - 1/D^{c'-4}$, no agent finds the target within $D^{2-f_2(D)}$ steps. With probability at least $1 - 1/D^{c'-3}$, R_0 steps correspond to at least one move. By a union bound, it follows that with probability at least $1 - 1/D^{c'-4}$, $\Delta = D^{2-f_2(D)}$ steps correspond to at least $\Delta/R_0 = D^{2-f_2(D)}/R_0 \geq D^{2-f_3(D)}$ moves. Therefore, no agent finds the target in $D^{2-f_3(D)}$ moves with probability at least $1 - 1/D^{c'-5} \geq 1 - 1/D^c$.

5.4 Theorem for M_{moves} and uniform algorithms

Finally, we extend Corollary 8 to uniform algorithms.

Corollary 9 *For any $D \in \mathbb{N}$, $D > 1$, any $n \in \mathbb{N}$, $n \leq T(D)$, and any uniform algorithm $\mathscr{A}$ with n agents, assume that $\chi(\mathscr{A}) = b + \log \ell \leq \log\log D - f_1(D)$. Then, there exists a placement (x,y), $|x|,|y| \leq D$ of the target, such that, for any constant $c > 1$, with probability at least $1 - 1/D^c$, algorithm $\mathscr{A}$ satisfies $M_{moves} > D^{2-f_3(D)}$ for this placement (x,y).*

Proof Note that the proofs of Theorem 6 and Corollary 8 are with respect to the Markov chain induced by the non-uniform algorithm. This Markov chain may have information about D encoded in it but throughout the proofs, the only way the value of D is used is through the constraint on the selection metric $\chi = b + \log \ell \leq \log\log D - f_1(D)$. Therefore, even if we consider a uniform algorithm, instead of a non-uniform algorithm, the results and the proofs still hold because the same restriction on χ applies to the uniform algorithm.

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

6 Summary and Future Work

We have presented algorithms 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 for n sub-exponential in D, no algorithm $\mathscr{A}$ can find the target with high probability in fewer than $D^{2-o(1)}$ rounds if $\chi(\mathscr{A}) < \log\log D - \omega(1)$. We have also presented two algorithms: (1) a non-uniform algorithm that finds the target in $\mathscr{O}(D^2/n + D)$ rounds in expectation for $\chi = \log\log D + \mathscr{O}(1)$, and (2) a uniform algorithm that finds the target in $\mathscr{O}((D + D^2/n)f(D))$ rounds in expectation and guarantees a χ-value of at most $2\log\log D + \mathscr{O}(1)$ for different choices of the function f. Our second algorithm also establishes a trade-off between the expected running time and the total number of bits used by the algorithm.

For future work, we consider various improvements to our algorithms. As mentioned earlier, we can make the algorithms also uniform in n by following the strategy in [14]; this modification will result in a $O(\log n)$ factor overhead in the running time. Furthermore, since our algorithms do not rely on communication or carefully synchronized rounds, it seems natural to analyze the fault tolerance properties the algorithms satisfy. We believe the correctness of the algorithms will not be affected by faults, as long as the faults are not adversarially targeted at a specific area of the grid; for example, if each agent that gets within distance of one hop to the target is crashed, then clearly our algorithms (or any other algorithms) cannot guarantee anything. Finally, it would also be interesting to consider various forms of communication between the agents and analyze the properties of the resulting algorithms. A recent attempt at understanding such behavior is [22], where the agents use only a loneliness detection capability in order to explore the grid with constant memory and constant probabilities.

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

Acknowledgements We express our gratitude to Yoav Rodeh and the anonymous reviewers who provided us with many insights, ideas how to strengthen our results and helped us improve the presentation of the material.

References

1. Afek, Y., Alon, N., Barad, O., Hornstein, E., Barkai, N., Bar-Joseph, Z.: A biological solution to a fundamental distributed computing problem. Science **331**(6014), 183–185 (2011)
2. Albers, S., Henzinger, M.R.: Exploring unknown environments. SIAM Journal on Computing **29**(4), 1164–1188 (2000)
3. Alon, N., Avin, C., Koucký, M., Kozma, G., Lotker, Z., Tuttle, M.R.: Many random walks are faster than one. Combinatorics, Probability and Computing **20**(04), 481–502 (2011)

4. Ambuhl, C., Gasieniec, L., Pelc, A., Radzik, T., Zhang, X.: Tree exploration with logarithmic memory. ACM Transactions on Algorithms **7**(2), 17 (2011)
5. Arbilly, M., Motro, U., Feldman, M.W., Lotem, A.: Co-evolution of learning complexity and social foraging strategies. Journal of Theoretical Biology **267**(4), 573–581 (2010)
6. Bender, M.A., Fernández, A., Ron, D., Sahai, A., Vadhan, S.: The power of a pebble: Exploring and mapping directed graphs. In: Proceedings of the ACM Symposium on Theory of Computing, pp. 269–278. ACM (1998)
7. Brauer, A.: On a problem of partitions. American Journal of Mathematics **64**(1), 299–312 (1942)
8. Deng, X., Papadimitriou, C.H.: Exploring an unknown graph. In: Proceedings of the Symposium on Foundations of Computer Science, pp. 355–361. IEEE (1990)
9. Diks, K., Fraigniaud, P., Kranakis, E., Pelc, A.: Tree exploration with little memory. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, pp. 588–597. Society for Industrial and Applied Mathematics (2002)
10. Emek, Y., Langner, T., Uitto, J., Wattenhofer, R.: Solving the ANTS problem with asynchronous finite state machines. In: Proceedings of the International Colloquium, pp. 471–482 (2014)
11. Emek, Y., Wattenhofer, R.: Stone age distributed computing. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 137–146. ACM (2013)
12. Feinerman, O., Korman, A.: Memory lower bounds for randomized collaborative search and implications for biology. In: Distributed Computing, pp. 61–75. Springer (2012)
13. Feinerman, O., Korman, A.: Theoretical distributed computing meets biology: A review. In: Distributed Computing and Internet Technology, pp. 1–18. Springer (2013)
14. Feinerman, O., Korman, A., Lotker, Z., Sereni, J.S.: Collaborative search on the plane without communication. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 77–86. ACM (2012)
15. Feller, W.: An introduction to probability theory and its applications, vol. 2. John Wiley & Sons (2008)
16. Fraigniaud, P., Gasieniec, L., Kowalski, D.R., Pelc, A.: Collective tree exploration. Networks **48**(3), 166–177 (2006)
17. Giraldeau, L.A., Caraco, T.: Social foraging theory. Princeton University Press (2000)
18. Harkness, R., Maroudas, N.: Central place foraging by an ant (Cataglyphis bicolor Fab.): a model of searching. Animal Behaviour **33**(3), 916–928 (1985)
19. Holder, K., Polis, G.: Optimal and central-place foraging theory applied to a desert harvester ant, Pogonomyrmex californicus. Oecologia **72**(3), 440–448 (1987)
20. Lenzen, C., Lynch, N., Newport, C., Radeva, T.: Trade-offs between selection complexity and performance when searching the plane without communication. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 252–261. ACM (2014)
21. McLeman, M., Pratt, S., Franks, N.: Navigation using visual landmarks by the ant leptothorax albipennis. Insectes Sociaux **49**(3), 203–208 (2002)
22. O'Brien, C.: Solving ANTS with Loneliness Detection and Constant Memory. MEng Thesis, MIT EECS Department (2014)
23. Panaite, P., Pelc, A.: Exploring unknown undirected graphs. Journal of Algorithms **33**(2), 281–295 (1999)
24. Reingold, O.: Undirected connectivity in log-space. Journal of the ACM (JACM) **55**(4), 17 (2008)
25. Robinson, E.J., Jackson, D.E., Holcombe, M., Ratnieks, F.L.: Insect communication: "no entry" signal in ant foraging. Nature **438**(7067), 442–442 (2005)
26. Rosenthal, J.S.: Rates of convergence for data augmentation on finite sample spaces. The Annals of Applied Probability pp. 819–839 (1993)

A Math Preliminaries

A.1 Basic Probability

In this section, we state a few basic concentration results.

Theorem 7 **(Chernoff bound)** *Let $X_1, \cdots, X_k$ be independent random variables such that for $1 \le i \le k$, $X_i \in \{0,1\}$. Let $X = X_1 + X_2 + \cdots + X_k$ and let $\mu = \mathbb{E}[X]$. Then, for any $0 \le \delta \le 1$, it is true that:*

$$P[X > (1+\delta)\mu] \le e^{-\delta^2\mu/2} \tag{7}$$

$$P[X < (1-\delta)\mu] \le e^{-\delta^2\mu/3} \tag{8}$$

Theorem 8 **(Two-sided Chernoff bound)** *Let $X_1, \cdots, X_k$ be independent random variables such that for $1 \le i \le k$, $X_i \in \{0,1\}$. Let $X = X_1 + X_2 + \cdots + X_k$ and let $\mu = \mathbb{E}[X]$. Then, for any $0 \le \delta \le 1$, it is true that:*

$$P[|X-\mu| > \delta\mu] \le 2e^{-\delta^2\mu/3} \tag{9}$$

A.2 Markov Chains

In this section, we state some basic results on Markov chains.

Theorem 9 **(Feller [15])** *In an irreducible Markov chain with period t the states can be divided into t mutually exclusive classes $G_0, \cdots, G_{t-1}$ such that it is true that (1) if $s \in G$ then the probability of being in state s in some round $r \ge 1$ is 0 unless $r = \tau + vt$ for some $v \in \mathbb{N}$, and (2) a one-step transition always leads to a state in the right neighboring class (in particular from G_{t-1} to G_0). In the chain with matrix P^t each class G corresponds to an irreducible closed set.*

The next theorem establishes a bound on the difference between the stationary distribution of a Markov chain and the distribution resulting after k steps.

Lemma 19 **(Rosenthal [26])** *Let $P(x,\cdot)$ be the transition probabilities for a time-homogeneous Markov chain on a general state space $\mathscr{X}$. Suppose that for some probability distribution $Q(\cdot)$ on $\mathscr{X}$, some positive integers k and k_0, and some $\varepsilon > 0$, $\forall x \in \mathscr{X} : P^{k_0}(x,\cdot) \ge \varepsilon Q(\cdot)$, where P^{k_0} represents the k_0-step transition probabilities. Then for any initial distribution π_0, the distribution π_k of the Markov chain after k steps satisfies $\|\pi_k - \pi\| \le (1-\varepsilon)^{\lfloor k/k_0 \rfloor}$, where $\|\cdot\|$ is the ∞-norm and π is any stationary distribution.*

Lemma 20 *In any irreducible, aperiodic Markov chain with $|S|$ states, there exists an integer $k \le 2|S|^2$ such that there is a walk of length k between any pair of states in the Markov chain.*

Proof By the definition of periodicity, for each state of the Markov chain, it is true that the greatest common divisor of the lengths of all the cycles that pass through that state is 1. Let the total number of distinct cycles in the Markov chain be m and let $(a_1, \cdots, a_m)$ denote the lengths of these cycles where $a_1 \le \cdots \le a_m$. The Frobenius number $F(a_1, \cdots, a_m)$ of the sequence $(a_1, \cdots, a_m)$ is the largest integer such that it is not possible to express it as a linear combination of $(a_1, \cdots, a_m)$ and non-negative integer coefficients. By a simple bound on the Frobenius number [7], we know that $F(a_1, \cdots, a_m) \le (a_1 - 1)(a_2 - 1) - 1$. Since a_1 and a_2 refer to cycle lengths in our Markov chain we know that $a_1, a_2 \le |S|$. So, it is true that $F(a_1, \cdots, a_m) \le |S|^2$ and we can express every integer greater than $F(a_1, \cdots, a_m)$ as a non-negative integer linear combination of $(a_1, \cdots, a_m)$.

Let i and j be arbitrary states in the Markov chain and let $d(i,j)$ be the shortest path between i and j. Let $k = 2|S|^2$. By the argument above, we know that there is a walk starting at state i and ending at state i of length $k - d(i,j) \ge |S|^2$. Appending the shortest path between i and j to the end of that walk results in a walk from i to j of length exactly k.