Memory Lower Bounds for Randomized Collaborative Search and Implications for Biology

Ofer Feinerman[1] ⋆ and Amos Korman[2] ⋆⋆

[1] Incumbent of the The Louis and Ida Rich Career Development Chair, The Weizmann Institute of Science, Rehovot, Israel. feiner@weizmann.ac.il
[2] CNRS and University Paris Diderot, France. amos.korman@liafa.jussieu.fr

Abstract. Initial knowledge regarding group size can be crucial for collective performance. We study this relation in the context of the *Ants Nearby Treasure Search (ANTS)* problem [18], which models natural cooperative foraging behavior such as that performed by ants around their nest. In this problem, k (probabilistic) agents, initially placed at some central location, collectively search for a treasure on the two-dimensional grid. The treasure is placed at a target location by an adversary and the goal is to find it as fast as possible as a function of both k and D, where D is the (unknown) distance between the central location and the target. It is easy to see that $T = \Omega(D + D^2/k)$ time units are necessary for finding the treasure. Recently, it has been established that $O(T)$ time is sufficient if the agents know their total number k (or a constant approximation of it), and enough memory bits are available at their disposal [18]. In this paper, we establish lower bounds on the agent memory size required for achieving certain running time performances. To the best our knowledge, these bounds are the first non-trivial lower bounds for the memory size of probabilistic searchers. For example, for every given positive constant ϵ, terminating the search by time $O(\log^{1-\epsilon} k \cdot T)$ requires agents to use $\Omega(\log \log k)$ memory bits.
From a high level perspective, we illustrate how methods from distributed computing can be useful in generating lower bounds for cooperative biological ensembles. Indeed, if experiments that comply with our setting reveal that the ants' search is time efficient, then our theoretical lower bounds can provide some insight on the memory ants use for this task.

1 Introduction

Background and Motivation: In biology, individuals assemble into groups that allow them, among other things, to monitor and react to relatively large environments. For this, the individuals typically spread over length scales that

⋆ Supported by the THE ISRAEL SCIENCE FOUNDATION FIRST grant (No. 1694/10) and by the Clore Foundation.
⋆⋆ Supported in part by the ANR projects DISPLEXITY and PROSE, and by the INRIA project GANG.

are much larger than those required for communication. Thus, collecting knowledge regarding large areas dictates a dispersion that may come at the price of group coordination. A possible solution involves the use of designated, localized areas where individuals convene to share information and from which they then disperse to interact with the environment. Indeed, such convention areas have been identified in a wide spectrum of biological systems ranging from groups of immuno-cells [33] to flocking birds [16,54].

Here we focus, on a third example, that of collective *central place foraging* [32,40] where a group of animals leave a central location (*e.g.,* a nest) to which they then retrieve collected food items. Ants, for example, were shown to communicate within their nest regarding food availability and quality outside it [9,29]. This information is then used as a means of regulating foraging efforts. It may be the case that such communication is constrained to the central place alone. For example, once they leave their nest, desert ants (*e.g.,* of the genus *Cataglyphys*) rarely interact [32]. This is due to their dispersedness and lack of chemical trail markings. Similar communication constraints are also experienced by the honeybee *Apis mellifera*. Other than being central-place, the search we discuss is cooperative: although it is conducted by individuals any findings are shared.

One piece of information that may be available to a localized group is its *size*. In a process known as *quorum sensing*, a threshold estimate of group size is used to reach collective decisions and choose between divergent courses of action [10,17,43,50,51]. Here, we study the potential benefits of estimating group size in the context of collective central place foraging without mid-search interactions. Clearly, due to competition and other time constrains, food items must be found relatively fast. Furthermore, finding food not only fast but also in proximity to the central location holds numerous advantages at both the search and the retrieval stages [32,38,40]. Intuitively, the problem at hand is efficiently distributing searchers within bounded areas around the nest while minimizing overlaps.

It was previously shown that the efficiency of collective central place foraging may be enhanced by initial knowledge regarding group size [18]. More specifically, that paper introduces the *Ants Nearby Treasure Search (ANTS)* problem, which models the aforementioned central place foraging setting. In this problem, k (probabilistic) agents, initially placed at some central location, collectively search for a treasure in the two-dimensional grid. The treasure is placed at a target location by an adversary and the goal is to find it as fast as possible as a function of both k and D, where D is the (unknown) distance between the central location and the target. Once the agents initiate the search they cannot communicate between themselves. Based on volume considerations, it is an easy observation that the expected running time of any algorithm is $\Omega(D + D^2/k)$. It was established in [18] that the knowledge of a constant approximation of k allows the agents to find the treasure in asymptotically optimal expected time, namely, $O(D + D^2/k)$. On the other hand, the lack of any information of k prevents them from reaching expected time that is higher than optimal by a

factor slightly larger than $O(\log k)$. That work also establishes lower bounds on the competitiveness of the algorithm in the particular case where some given approximation to k is available to all nodes.

In this work, we simulate the initial step of information sharing (*e.g.,* regarding group size) within the nest by using the abstract framework of *advice* (see, *e.g.,* [12,23,25]). That is, we model the preliminary process for gaining knowledge about k (*e.g.,* at the central location) by means of an *oracle* that assigns advice to agents. To measure the amount of information accessible to agents, we analyze the *advice size*, that is, the maximum number of bits used in an advice. Since we are mainly interested in lower bounds on the advice size required to achieve a given competitive ratio, we apply a liberal approach and assume a highly powerful oracle. More specifically, even though it is supposed to model a distributed (probabilistic) process, we assume that the oracle is a centralized probabilistic algorithm (almost unlimited in its computational power) that can assign different agents different advices. Note that, in particular, by considering identifiers as part of the advice, our model allows to relax the assumption that all agents are identical and to allow agents to be of several types. Indeed, in the context of ants, it has been established that ants on their first foraging bouts execute different protocols than those that are more experienced [52].

The main technical results of this paper deal with lower bounds on the advice size. For example, with the terminology of advice, [18] showed that advice of size $O(\log \log k)$ bits is sufficient to obtain an $O(1)$-competitive algorithm. We prove that this bound is tight. In fact, we show a much stronger result, that is, that advice of size $\Omega(\log \log k)$ is necessary even for achieving competitiveness which is as large as $O(\log^{1-\epsilon} k)$, for every given positive constant ϵ. On the other extremity, we show that $\Omega(\log \log \log k)$ bits of advice are necessary for being $O(\log k)$-competitive, and that this bound is tight. In addition, we exhibit lower bounds on the corresponding advice size for a range of intermediate competitivenesses.

Observe that the advice size bounds from below the number of memory bits used by an agent, as this amount of bits in required merely for storing some initial information.

In general, from a purely theoretical point of view, analyzing the memory required for efficient search is a central theme in computer science [45], and is typically considered to be difficult. To the best of our knowledge, the current paper is the first paper establishing non-trivial lower bounds for the memory of randomized searching agents with respect to given time constrains.

From a high level perspective, we illustrate that distributed computing can potentially provide a novel and efficient methodology for the study of highly complex, cooperative biological ensembles. Indeed, if experiments that comply with our setting reveal that the ants' search is time efficient, in the sense detailed above, then our theoretical results can provide some insight on the memory ants use for this task. A detailed discussion of this novel approach is given in Section 5.

Our results: The main technical results deal with lower bounds on the advice size. Our first result is perhaps the most surprising one. It says not only that

$\Omega(\log\log k)$ bits of advice are required to obtain an $O(1)$-competitive algorithm, but that roughly this amount is necessary even for achieving competitiveness which is as large as $O(\log^{1-\epsilon} k)$, for every given positive constant ϵ. This result should be put in contrast to the fact that with no advice at all, one can obtain a search algorithm whose competitiveness is slightly higher than logarithmic [18].

Theorem 1. *There is no search algorithm that is $O(\log^{1-\epsilon} k)$-competitive for some fixed positive ϵ, using advice of size $o(\log\log k)$.*

On the other extremity, we show that $\Omega(\log\log\log k)$ bits of advice are necessary for constructing an $O(\log k)$-competitive algorithm, and we prove that this bound on the advice is in fact tight.

Theorem 2. *There is no $O(\log k)$-competitive search algorithm, using advice of size $\log\log\log k - \omega(1)$. On the other hand, there exists an $O(\log k)$-competitive search algorithm using advice of size $\log\log\log k + O(1)$.*

Finally, we also exhibit lower bounds for the corresponding advice size for a range of intermediate competitivenesses.

Theorem 3. *Consider a $\Phi(k)$-competitive search algorithm using advice of size $\Psi(k)$. Then, $\Phi(k) = \Omega(\log k/2^{\Psi(k)})$, or in other words, $\Psi(k) = \log\log k - \log\Phi(k) - O(1)$. In particular, if $\Phi(k) = \frac{\log k}{2^{\log^{\epsilon}\log k}}$, then $\Psi(k) = \log^{\epsilon}\log k - O(1)$.*

Our results on the advice complexity are summarized in Table 1. As mentioned, our lower bounds on the advice size are also lower bounds on the memory size of agents.

	Competitiveness	Advice size
Tight bound	$O(1)$	$\Theta(\log\log k)$
Tight bound	$O(\log^{1-\epsilon} k)$ $\quad 0 < \epsilon < 1$	$\Theta(\log\log k)$
Lower bound	$\log k/2^{\log^{\epsilon}\log k}$ $\quad 0 < \epsilon < 1$	$\log^{\epsilon}\log k - O(1)$
Tight bound	$O(\log k)$	$\log\log\log k + \Theta(1)$
Upper bound [18]	$O(\log^{1+\epsilon} k)$	zero

Table 1: Bounds on the advice for given competitiveness

Related Work: Our current work falls within the framework of natural algorithms, a recent attempt to study biological phenomena from an algorithmic perspective [1,8,11,18].

The notion of advice is central in computer science. In particular, the concept of advice and its impact on various computations has recently found various applications in distributed computing. In this context, the main measure used

is the advice size. It is for instance analyzed in frameworks such as proof labeling [36,37], broadcast [23], local computation of MST [25], graph coloring [24] and graph searching by a single robot [12,31]. Very recently, it has also been investigated in the context of online algorithms [15].

Collective search is a classical problem that has been extensively studied in different contexts (for a more comprehensive summary refer to [18]). Social foraging theory [28] and central place foraging typically deal with optimal resource exploitation strategies between competing or cooperating individuals. Actual collective search trajectories of non-communicating agents have been studied in the physics literature (*e.g.,* [6,46]). Reynolds [46] achieves optimal speed up through overlap reduction which is obtained by sending searchers on near -straight disjoint lines to infinity. This must come at the expense of finding proximal treasures. Harkness and Maroudas [32] combined field experiments with computer simulations of a semi-random collective search and suggest substantial speed ups as group size increases. The collective search problem has further been studied from an engineering perspective (*e.g.,* [42]). In this case, the communication between agents (robots) or their computational abilities are typically unrestricted. These works put no emphasis on finding nearby treasures fast. Further, there is typically (with the exception of [32]) no reference to group size.

In the theory of computer science, the exploration of graphs using mobile agents (or robots) is a central question. (For a more detailed survey refer to *e.g.,* [2,18,20].) Most graph exploration research in concerned with the case of a single deterministic agent exploring a finite graph, see for example [3,13,14,27,41,45]. The more complex situation of multiple identical deterministic agents was studied in [4,20,21,22]. In general, one of the main challenges in search problems is the establishment of memory bounds. For example, the question of whether a single agent can explore all finite undirected graphs using logarithmic memory was open for a long time; answering it to the affirmative [45] established an equality between the classes of languages SL and L.

Evaluating the running time as a function of D, the distance to the treasure, was studied in the context of the cow-path problem [5,34]. In [39], the cow-path problem was extended by considering k agents. However, in contrast to our setting, the agents they consider have unique identities, and the goal is achieved by (centrally) specifying a different path for each of the k agents.

The question of how important it is for individual processors to know their total number has recently been addressed in the context of locality. Generally speaking, it has been observed that for several classical local computation tasks, knowing the number of processors is not essential [35]. On the other hand, in the context of local distributed decision, some evidence exist that such knowledge is crucial for non-deterministic verification [26].

2 Preliminaries

General setting: We consider the *Ants Nearby Treasure Search (ANTS)* problem initially introduced in [18]. In this *central place* searching problem, k mobile

agents are searching for a *treasure* on the two-dimensional plane. The agents are probabilistic mobile machines (robots). They are identical, that is, all agents execute the same protocol $\mathcal{P}$. Each agent has some limited field of view, i.e., each agent can see its surrounding up to a distance of some $\varepsilon > 0$. Hence, for simplicity, instead of considering the two-dimensional plane, we assume that the agents are actually walking on the integer two-dimensional infinite grid $G = \mathbb{Z}^2$ (they can traverse an edge of the grid in both directions). The search is central place, that is, all k agents initiate the search from some central node $s \in G$, called the *source*. Before the search is initiated, an adversary locates the treasure at some node $t \in G$, referred to as the *target* node. Once the search is initiated, the agents cannot communicate among themselves[3]. We denote by D the (Manhattan) distance between the source node and the target, i.e., $D = d_G(s, t)$. It is important to note that the agents have no a priori information about the location of t or about D. We say that the agents *find* the treasure when one of the agents visits the target node t. The goal of the agents it to find the treasure as fast as possible as a function of both D and k.

Since we are mainly interested in lower bounds, we assume a very liberal setting. In particular, we do not restrict neither the computational power nor the navigation capabilities of agents. Moreover, we put no restrictions on the internal storage used for navigation. (On the other hand, we note that for constructing upper bounds, the algorithms we consider use simple procedures that can be implemented using relatively little resources.) In addition, even though it may seem natural to require that agents return to the source occasionally (*e.g.,* to know whether other agents have already found the treasure), our lower bounds do not rely on such an assumption.

Oracles and Advice: We would like to model the situation in which before the search actually starts, some initial communication may be made between the agents at the source node. In reality, this preliminary communication may be quite limited. This may be because of difficulties in the communication that are inherent to the agents or the environment, *e.g.,* due to faults or limited memory, or because of asynchrony issues regarding the different starting times of the search, or simply because agents are identical and it may be difficult for agents to distinguish one agent from the other. Nevertheless, we consider a very liberal setting in which this preliminary communication is almost unrestricted.

More specifically, we consider a centralized algorithm called *oracle* that assigns advices to agents in a preliminary stage. The oracle, denoted by $\mathcal{O}$, is a probabilistic centralized algorithm that receives as input a set of k agents and assigns an *advice* to each of the k agents. We assume that the oracle may use a different protocol for each k; given k, the randomized algorithm used for assign-

[3] As mentioned earlier, the particular kinds of animals that we are interested in (*e.g.,* the desert ants *Cataglyphys* and the honeybees *Apis mellifera*) rarely interact outside their source while searching [32]. Nevertheless, the theoretical question of how and to what extent mid-search interactions may aid the search is interesting and a subject of future work.

ing the advices to the k agents is denoted by $\mathcal{O}_k$. An example for such an oracle is the case where each agent is given the same (deterministic) advice which encodes some approximation to the number of agents[4]. However, in principle, the oracle may assign a different advice to each agent, and these advices may not necessarily correspond to an approximation of k. Observe, this definition of an oracle allows it to simulate almost any reasonable preliminary communication between the agents[5].

It is important to stress that even though all agents execute the same searching protocol, they may start the search with different advices. Hence, since their searching protocol may rely on the content of this initial advice, agents with different advices may behave differently. Another important remark concerns the fact that some part of the advices may be used for encoding (not necessarily disjoint) identifiers. That is, assumptions regarding the settings in which not all agents are identical and there are several types of agents can be captured by our setting of advice.

Finally, note that, in contrast to previous works regarding oracles, here, the knowledge of the oracle is limited, as it does not know D. This means, in particular, that a probabilistic oracle may potentially be strictly more powerful than a deterministic one. Indeed, the oracle assigning the advice may use the randomization to reduce the size of the advices by balancing between the efficiency of the search for small values of D and larger values.

To summarize, a *search algorithm* is a pair $\langle \mathcal{P}, \mathcal{O} \rangle$ consisting of a randomized searching protocol $\mathcal{P}$ and randomized oracle $\mathcal{O} = \{\mathcal{O}_k\}_{k \in \mathbf{N}}$. Given k agents, the randomized oracle $\mathcal{O}_k$ assigns a separate advice to each of the given agents. Subsequently, all agents initiate the actual search by letting each of the agents execute protocol $\mathcal{P}$ and using the corresponding advice as input to $\mathcal{P}$. Once the search is initiated, the agents cannot communicate among themselves.

Consider an oracle $\mathcal{O}$. Given k, let $\Psi_{\mathcal{O}}(k)$ denote the maximum number of bits devoted for encoding the advice of an agent, taken over all coin tosses of $\mathcal{O}_k$, and over the k agents. In other words, $\Psi_{\mathcal{O}}(k)$ is the minimum number of bits necessary for encoding the advice, assuming the number of agents is k. Note that $\Psi_{\mathcal{O}}(k)$ also bounds from below the number of memory bits of an agent required by the search algorithm $\langle \mathcal{P}, \mathcal{O} \rangle$, assuming that the number of agents is k. The function $\Psi_{\mathcal{O}}(\cdot)$ is called the *advice size* function of oracle $\mathcal{O}$. (When the context is clear, we may omit the subscript $\mathcal{O}$ from $\Psi_{\mathcal{O}}(\cdot)$ and simply use $\Psi(\cdot)$ instead.)

---

[4] These types of simple oracles are actually used by our upper bound constructions.

[5] For example, it can simulate to following very liberal setting. Assume that in the preprocessing stage, the k agents are organized in a clique topology, and that each agent can send a separate message to each other agent. Furthermore, even though the agents are identical, in this preprocessing stage, let us assume that agents can distinguish the messages received from different agents, and that each of the k agents may use a different probabilistic protocol for this preliminary communication. In addition, no restriction is made neither on the memory and computation capabilities of agents nor on the preprocessing time, that is, the preprocessing stage takes finite, yet unlimited, time.

Time complexity: When measuring the time to find the treasure, we assume that all internal computations are performed in zero time. For the simplicity of presentation, we assume that the movements of agents are synchronized, that is, each edge traversal is performed in precisely one unit of time. Indeed, this assumption can easily be removed if we measure the time according to the slowest edge-traversal. We also assume that all agents start the search simultaneously at the same time. This assumption can also be easily removed by starting to count the time when the last agent initiates the search.

The *expected running time* of a search algorithm $\mathcal{A} := \langle \mathcal{P}, \mathcal{O} \rangle$ is the expected time until at least one of the agents finds the treasure. The expectation is defined with respect to the coin tosses made by the (probabilistic) oracle $\mathcal{O}$ assigning the advices to the agents, as well as the subsequent coin tosses made by the agents executing $\mathcal{P}$. We denote the expected running time of an algorithm $\mathcal{A}$ by $\tau = \tau_{\mathcal{A}}(D, k)$. In fact, for our lower bound to hold, it is sufficient to assume that the probability that the treasure is found by time 2τ is at least $1/2$. By Markov inequality, this assumption is indeed weaker than the assumption that the expected running time is τ.

Note that if an agent knows D, then it can potentially find the treasure in time $O(D)$, by walking to a distance D in some direction, and then performing a circle around the source of radius D (assuming, of course, that its navigation abilities enable it to perform such a circle). On the other hand, with the absence of knowledge about D, an agent can find the treasure in time $O(D^2)$ by performing a spiral search around the source (see, *e.g.,* [5]). The following observation imply that $\Omega(D + D^2/k)$ is a lower bound on the expected running time of any search algorithm. The proof is straightforward and can be found in [18].

Observation 4 *The expected running time of any algorithm is $\Omega(D + D^2/k)$, even if the number of agents k is known to all agents.*

We evaluate the time performance of an algorithm with respect to the lower bound given by Observation 4. Formally, let $\Phi(k)$ be a function of k. A search algorithm $\mathcal{A} := \langle \mathcal{P}, \mathcal{O} \rangle$ is called $\Phi(k)$-*competitive* if $\tau_{\mathcal{A}}(D, k) \leq \Phi(k) \cdot (D + D^2/k)$, for every integers k and D. Our goal is establish connections between the *size* of the advice, namely $\Psi(k)$, and the competitiveness $\Phi(k)$ of the search algorithm.

More definitions: The *distance* between two nodes $u, v \in G$, denoted $d(u, v)$, is simply the Manhattan distance between them, i.e., the number of edges on the shortest path connecting u and v in the grid G. For a node u, let $d(u) := d(u, s)$ denote the distance between u and the source node. Hence, $D = d(t)$.

3 Lower Bounds on the Advice

The theorem below generalizes Theorem 4.1 in [18], taking into account the notion of advice. All our lower bound results follow as corollaries of this theorem. Note that for the theorem to be meaningful we are interested in advice size whose order of magnitude is less than $\log \log k$. Indeed, if $\Psi(k) = \log \log k$, then one

can encode a 2-approximation of k in each advice, and obtain an optimal result, that is, an $O(1)$-competitive algorithm (see [18]).

Before stating the theorem, we need the following definition. A non-decreasing function $\Phi(x)$ is called *relatively-slow* if $\Phi(x)$ is sublinear (i.e., $\Phi(x) = o(x)$) and if there exist two positive constants c_1 and $c_2 < 2$ such that when restricted to $x > c_1$, we have $\Phi(2x) < c_2 \cdot \Phi(x)$. Note that this definition captures many natural sublinear functions[6].

Theorem 5. *Consider a $\Phi(k)$-competitive search algorithm using advice of size $\Psi(k)$. Assume that $\Phi(\cdot)$ is relatively-slow and that $\Psi(\cdot)$ is non-decreasing. Then there exists some constant x', such that for every $k > 2^{x'}$, the sum $\sum_{i=x'}^{\log k} \frac{1}{\Phi(2^i) \cdot 2^{\Psi(k)}}$ is at most some fixed constant.*

Proof. Consider a search algorithm with advice size $\Psi(k)$ and competitiveness $\Phi'(k)$, where $\Phi'(\cdot)$ is relatively-slow. By definition, the expected running time is less than $\tau(D,k) = (D + D^2/k) \cdot \Phi'(k)$. Note, for $k \leq D$, we have $\tau(D,k) \leq \frac{D^2 \Phi(k)}{k}$, where $\Phi(k) = 2\Phi'(k)$, and $\Phi(\cdot)$ is relatively-slow. Let c_1 be the constant promised by the fact that Φ is relatively-slow. Let $x_0 > c_1$ be sufficiently large so that x_0 is a power of 2, and for every $x > x_0$, we have $\Phi(x) < x$ (recall, Φ is sublinear).

Fix an integer $T > x_0^2$. In the remaining of the proof, we assume that the treasure is placed somewhere at distance $D := 2T + 1$. Note, this means, in particular, that by time $2T$ the treasure has not been found yet.

Fix an integer i in $[\log x_0, \frac{1}{2} \log T]$, set $d_i = \sqrt{\frac{T \cdot k_i}{\Phi(k_i)}}$, and let $B(d_i) := \{v \in G : d(v) \leq d_i\}$ denote the ball of radius d_i around the source node. We consider now the case where the algorithm is executed with $k_i := 2^i$ agents (using the corresponding advices given by the oracle $\mathcal{O}_{k_i}$). For every set of nodes $S \subseteq B(d_i)$, let $\chi_i(S)$ denote the random variable indicating the number of nodes in S that were visited by at least one of the k_i agents by time $2T$. (For short, for a singleton node u, we write $\chi_i(u)$ instead of $\chi_i(\{u\})$.) Note, the value of $\chi_i(S)$ depends on the values of the coins tosses made by the oracle for assigning the advices as well as on the values of the coins tossed by the k_i agents. Now, define the ring

$$R_i := B(d_i) \setminus B(d_{i-1}).$$

Claim. For each integer $i \in [\log x_0, \frac{1}{2} \log T]$, we have $\mathbf{E}(\chi_i(R_i)) = \Omega(d_i^2)$.

To see why the claim holds, note that by the properties of Φ, and from the fact that $2^i \leq \sqrt{T}$, we get that $k_i \leq d_i$, and therefore, $\tau(d_i, k_i) \leq \frac{d_i^2 \Phi(k_i)}{k_i} = T$. It follows that for each node $u \in B(d_i)$, we have $\tau(d(u), k_i) \leq T$, and hence, the probability that u is visited by time $2T$ is at least $1/2$, that is,

[6] For example, note that the functions of the form $\alpha_0 + \alpha_1 \log^{\beta_1} x + \alpha_2 \log^{\beta_2} \log x + \alpha_3 2^{\log^{\beta_3} \log x} \log x + \alpha_4 \log^{\beta_4} x \log^{\beta_5} \log x$, (for non-negative constants α_i and β_i, $i = 1,2,3,4,5$ such that $\sum_{i=1}^{4} \alpha_i > 0$) are all relatively-slow.

$\mathbf{Pr}(\chi_i(u) = 1) \geq 1/2$. Hence, $\mathbf{E}(\chi_i(u)) \geq 1/2$. Now, by linearity of expectation, $\mathbf{E}(\chi_i(R_i)) = \sum_{u \in R_i} \mathbf{E}(\chi_i(u)) \geq |R_i|/2$. Consequently, by time $2T$, the expected number of nodes in R_i that are visited by the k_i agents is $\Omega(|R_i|) = \Omega\left(d_{i-1}(d_i - d_{i-1})\right) = \Omega\left(\frac{T \cdot k_i}{\Phi(k_{i-1})} \cdot \left(\sqrt{\frac{2\Phi(k_{i-1})}{\Phi(k_i)}} - 1\right)\right) = \Omega\left(\frac{T \cdot k_i}{\Phi(k_i)}\right) = \Omega(d_i^2)$, where the second equality follows from the fact that $d_i = d_{i-1} \cdot \sqrt{\frac{2\Phi(k_{i-1})}{\Phi(k_i)}}$, and the third equality follows from the fact that $\Phi(\cdot)$ is relatively-slow. This establishes the claim.

Note that for each $i \in [\log x_0 + 1, \frac{1}{2}\log T]$, the advice given by the oracle to any of the k_i agents must use at most $\Psi(k_i) \leq \Psi(\sqrt{T})$ bits. In other words, for each of these k_i agents, each advice is some integer whose value is at most $2^{\Psi(\sqrt{T})}$.

Let $W(j,i)$ denote the random variable indicating the number of nodes in R_i visited by the j'th agent by time $2T$, assuming that the total number of agents is k_i. By Claim 3, for every integer $i \in [\log x_0 + 1, \frac{1}{2}\log T]$, we have: $\mathbf{E}(\sum_{j=1}^{k_i} W(j,i)) \geq \mathbf{E}(\chi_i(R_i)) = \Omega(d_i^2)$. By linearity of expectation, it follows that for every integer $i \in [\log x_0 + 1, \frac{1}{2}\log T]$, there exists an integer $j \in \{1, 2, \cdots, k_i\}$ for which $\mathbf{E}(W(j,i)) = \Omega(d_i^2/k_i) = \Omega(T/\Phi(k_i))$.

Now, for each advice in the relevant range, i.e., for each $a \in \{1, \cdots, 2^{\Psi(\sqrt{T})}\}$, let $M(a,i)$ denote the random variable indicating the number of nodes in R_i that an agent with advice a visits by time $2T$. Note, the value of $M(a,i)$ depends only on the values of the coin tosses made by the agent. On the other hand, note that the value of $W(j,i)$ depends on the results of the coin tosses made by the oracle assigning the advice, and the results of the coin tosses made by the agent that uses the assigned advice. Recall, the oracle may assign an advice to agent j according to a distribution that is different than the distributions used for other agents. However, regardless of the distribution used by the oracle for agent j, it must be the case that there exists an advice $a_i \in \{1, \cdots, 2^{\Psi(\sqrt{T})}\}$, for which $\mathbf{E}(M(a_i,i)) \geq \mathbf{E}(W(j,i))$. Hence, we obtain:

$$\mathbf{E}(M(a_i,i)) = \Omega(T/\Phi(k_i)).$$

Let $A = \{a_i \mid i \in [\log x_0 + 1, \frac{1}{2}\log T]\}$. Consider now an "imaginary" scenario[7] in which we execute the search algorithm with $|A|$ agents, each having a different advice in A. That is, for each advice $a \in A$, we have a different agent executing the algorithm using advice a. For every set S of nodes, let $\hat{\chi}(S)$ denote the random variable indicating the number of nodes in S that were visited by at least one of these $|A|$ agents (in the "imaginary" scenario) by time $2T$. Let

[7] The scenario is called imaginary, because, instead of letting the oracle assign the advice for the agents, we impose a particular advice to each agent, and let the agents perform the search with our advices. Note, even though such a scenario cannot occur by the definition of the model, each individual agent with advice a cannot distinguish this case from the case that the number of agents was some k' and the oracle assigned it the advice a.

$\hat{\chi} := \hat{\chi}(G)$ denote the random variable indicating the total number of nodes that were visited by at least one of these agents by time $2T$.

By definition, for each $i \in [\log x_0 + 1, \frac{1}{2}\log T]$, the expected number of nodes in R_i visited by at least one of these $|A|$ agents is $\mathbf{E}(\hat{\chi}(R_i)) \geq \mathbf{E}(M(a_i, i)) = \Omega(T/\Phi(k_i))$. Since the sets R_i are pairwise disjoint, the linearity of expectation implies that the expected number of nodes covered by these agents by time $2T$ is $\mathbf{E}(\hat{\chi}) \geq \sum_{i=x_0+1}^{\frac{1}{2}\log T} \mathbf{E}(\hat{\chi}(R_i)) = \Omega\left(\sum_{i=x_0+1}^{\frac{1}{2}\log T} \frac{T}{\Phi(k_i)}\right) = T \cdot \Omega\left(\sum_{i=x_0+1}^{\frac{1}{2}\log T} \frac{1}{\Phi(2^i)}\right)$. Recall that A is included in $\{1, \cdots, 2^{\Psi(\sqrt{T})}\}$. Hence, once more by linearity of expectation, there must exist an advice $\hat{a} \in A$, such that the expected number of nodes that an agent with advice $\hat{a}$ visits by time $2T$ is $T \cdot \Omega\left(\sum_{i=x_0+1}^{\frac{1}{2}\log T} \frac{1}{\Phi(2^i) \cdot 2^{\Psi(\sqrt{T})}}\right)$. Since each agent may visit at most one node in one unit of time, it follows that, for every T large enough, the sum $\sum_{i=x_0+1}^{\frac{1}{2}\log T} 1/\Phi(2^i) \cdot 2^{\Psi(\sqrt{T})}$ is at most some fixed constant. The proof of the theorem now follows by replacing the variable T with T^2. □

Corollary 1. *Consider a $\Phi(k)$-competitive search algorithm using advice of size $\Psi(k)$. Assume that $\Phi(\cdot)$ is relatively-slow. Then, $\Phi(k) = \Omega(\log k/2^{\Psi(k)})$, or in other words, $\Psi(k) = \log\log k - \log\Phi(k) - O(1)$.*

Proof. Theorem 5 says that for every k, we have $\frac{1}{2^{\Psi(k)}} \sum_{i=1}^{\log k} \frac{1}{\Phi(2^i)} = O(1)$. On the other hand, since Φ is non-decreasing, we have $\sum_{i=1}^{\log k} \frac{1}{\Phi(2^i)} \geq \frac{\log k}{\Phi(k)}$. Hence, $\frac{\log k}{2^{\Psi(k)} \cdot \Phi(k)} = O(1)$. The corollary follows. □

The following corollary follows directly from the previous one.

Corollary 2. *Let $\epsilon < 1$ be a positive constant. Consider a $\frac{\log k}{2^{\log^\epsilon \log k}}$-competitive search algorithm using advice of size $\Psi(k)$. Then $\Psi(k) = \log^\epsilon \log k - O(1)$.*

Our next corollary implies that even though $O(\log\log k)$ bits of advice are sufficient for obtaining $O(1)$-competitiveness, roughly this amount of advice is necessary even for achieving relatively large competitiveness.

Corollary 3. *There is no $O(\log^{1-\epsilon} k)$-competitive search algorithm for some positive constant ϵ, using advice of size $\Psi(k) = \epsilon \log\log k - \omega(1)$.*

Proof. Assume that the competitiveness is $\Phi(k) = O(\log^{1-\epsilon} k)$. Then, we have $\sum_{i=1}^{\log k} \frac{1}{\Phi(2^i) \cdot 2^{\Psi(k)}} = \Omega\left(\frac{\log^\epsilon k}{2^{\Psi(k)}}\right)$. According to Theorem 5, this sum is constantly bounded, and hence, we cannot have $\Psi(k) = \epsilon \log\log k - \omega(1)$. □

Corollary 4. *There is no $O(\log k)$-competitive search algorithm, using advice of size $\log\log\log k - \omega(1)$.*

Proof. Assume that the competitiveness is $\Phi(k) = O(\log k)$. Since $\Phi(2^i) = O(i)$, we have $\sum_{i=1}^{\log k} 1/\Phi(2^i) = \sum_{i=1}^{\log k} 1/i = \Omega(\log\log k)$. According to Theorem 5, $\frac{1}{2^{\Psi(k)}} \sum_{i=1}^{\log k} 1/\Phi(2^i) = \Omega(\log\log k/2^{\Psi(k)})$ must converge as k goes to infinity. In particular, we cannot have $\Psi(k) = \log\log\log k - \omega(1)$. □

4 Upper Bound

The lower bound on the advice size given in Corollary 3 is tight, as $O(\log \log k)$ bits of advice are sufficient to obtain an $O(1)$-competitive search algorithm. To further illustrate the power of Theorem 5, we now claim that the lower bound mentioned in Corollary 4 is also tight. Theorem 2 follows by combining the Theorem 6 below and Corollary 4. Due to space considerations, the proof of Theorem 6 is deferred to the full version of this paper.

Theorem 6. *There exists an* $O(\log k)$*-competitive algorithm using* $\log \log \log k + O(1)$ *bits of advice.*

5 Implications for Biology

A common problem, when studying a biological system is the complexity of the system and the huge number of parameters involved. This raises the need for more concise descriptions and several alternatives have been explored. One tactic is reducing the parameter space. This is done by dividing the parameter space into critical and non-critical directions where changes in non-critical parameters do not affect overall system behavior [19,30]. A different approach involves the definitions of bounds which govern biological systems independently of any algorithms or parameters. Previous works have utilized physics [7] and information theory [44] to set such bounds.

Our results are an attempt to draw non-trivial bounds on biological systems from the field of *distributed computing*. Such theoretical lower bounds on advice size may enable one to relate group search performance to the extent of information sharing within the nest. These types of bounds are particularly interesting since they provide not a single rule but relations between key parameters. In our case these would be the memory capacity of an agent and collective search efficiency.

We do not claim that our setting precisely captures the framework in which the aforementioned species perform search, yet we do believe that it provides a first approximation for it. Indeed, apart from engaging in central-place foraging with no mid-search communication, these species possess many of the individual skills required for the behavioral patterns that are utilized in our upper bounds [32,47,48,49,52,53]. It is not unreasonable to assume that a careful inspection of these species in nature would reveal a slightly different framework and would require the formulation of similar suitable theoretical memory bounds. Combining such memory lower bounds with experimental measurements of search speed with varying numbers of searchers would then provide quantitative evidence regarding the number of memory bits (or, alternatively, the number of states) used by ants during a search. In particular, this would help to understand the ants' quorum sensing process, as this number of memory bits are required merely for representing the output of that process. Although such experiments are beyond the scope of the current work, our results provide a "proof-of-concept" for this methodology.

References

1. Y. Afek, N. Alon, O. Barad, E. Hornstein, N. Barkai, and Z. Bar-Joseph. A biological solution to a fundamental distributed computing problem. Science, 331(6014):183–185, 2011.
2. S. Alpern and S. Gal. The Theory of Search Games and Rendezvous. *Kluwer (now Springer) Academic Publishers*, 2003, 319 P.
3. S. Albers and M. R. Henzinger. Exploring unknown environments. *SIAM J. on Computing*, (2000), 29, 1164–1188.
4. I. Averbakh and O. Berman. $(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective. *Discr. Appl. Mathematics*, (1997), 75, 201–216.
5. R. A. Baeza-Yates, J. C. Culberson and G.J.E. Rawlins. Searching in The Plane. *Information and Computation*, (1991), (106), (2), 234–252.
6. G. Berkolaiko and S. Havlin Territory covered by N Levy flights on d-dimensional lattices. Physical review. E 55(2): 1395–1400, 1999.
7. W. Bialek. Physical limits to sensation and perception. Annual Review of Biophysics and Biophysical Chemistry, 16:455–478, 1987.
8. V. Bonifaci, K. Mehlhorn, G. Varma. Physarum can compute shortest paths. *Proc. 23th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2012, pages 233–240.
9. J. Le Breton and V. Fourcassié. Information transfer during recruitment in the ant Lasius niger L. (Hymenoptera: Formicidae). Behavioral Ecology and Sociobiology 55(3):242–250, 2004.
10. N.J. Burroughs, M.de.O.B. Miguel Paz B and P.A. Adrego. Regulatory Tcell adjustment of quorum growth thresholds and the control of local immune responses. J. of Theoretical Biology 241:134–141, 2006.
11. B. Chazelle. Natural algorithms. SODA 2009, pages 422–431.
12. R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman and D. Peleg. Label-Guided Graph Exploration by a Finite Automation. ACM Transactions on Algorithms (TALG) 4(4), 2008.
13. A. Dessmark and A. Pelc. Optimal graph exploration without good maps. In ESA 2002, LNCS 2461, 374–386.
14. K. Diks, P. Fraigniaud, E. Kranakis and A. Pelc. Tree exploration with little memory. In SODA 2002, 588–597.
15. Y. Emek, P. Fraigniaud, A. Korman and A. Rosen. Online Computation with Advice. Theoretical Computer Science (TCS) , 412(24), (2011), 2642–2656.
16. C.J. Feare, G.M. Dunnet and I.J. Patterson. Ecologicalstudies of the rook (*Corvus frugilegus L.*) in north-east Scotland; Food intake and feeding behaviour. J. of Applied Ecology 11: 867–896, 1974.
17. O. Feinerman, G. Jentsch, K.E. Tkach, J.W. Coward, M.M. Hathorn, M.W. Sneddon, T. Emonet, K.A. Smith and G. Altan-Bonnet. Single-cell quantification of IL-2 response by effector and regulatory T cells reveals critical plasticity in immune response. Molecular systems biology. 6(437), 2010.
18. O. Feinerman, A. Korman, Z. Lotker and J.S. Sereni. Collaborative Search on the Plane without Communication. To appear in PODC 2012.
19. O. Feinerman, J. Veiga, J.R. Dorfman, R.N. Germain and G. Altan-Bonnet. Variability and robustness in T Cell activation from regulated heterogeneity in protein levels. Science 321(5892): 1081–1084, 2008.

20. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Remembering without memory: tree exploration by asynchronous oblivious robots. TCS, 411, 1583–1598, 2010.
21. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. How many oblivious robots can explore a line. *Inf. Process. Lett* (2011), 111(20): 1027–1031.
22. P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective tree exploration. In LATIN 2004, LNCS 2976, 141–151.
23. P. Fraigniaud, D. Ilcinkas, and A. Pelc. Oracle size: a new measure of difficulty for communication tasks. In PODC 2006, pages 179–187.
24. P. Fraigniaud, C. Gavoille, D. Ilcinkas, and A. Pelc. Distributed computing with advice: information sensitivity of graph coloring. In ICALP 2007.
25. P. Fraigniaud, A. Korman and E. Lebhar. Local MST Computation with Short Advice. Theory of Computing Systems (ToCS) , 47(4), (2010), 920–933.
26. P. Fraigniaud, A. Korman, and D. Peleg. Local Distributed Decision. In FOCS 2011.
27. L. Gasieniec, A. Pelc, T. Radzik, X. Zhang. Tree exploration with logarithmic memory. In SODA 2007.
28. L.A. Giraldeau and T. Carco. Social Foraging Theory, 2000.
29. D.M. Gordon. The regulation of foraging activity in red harvester ant colonies. The American Naturalist 159(5):509–518 ,2002.
30. R.N. Gutenkunst, J.J. Waterfall, F.P. Casey, K.S. Brown, C.R. Myers and J.P. Sethna. Universally sloppy parameter sensitivities in systems biology models. PLOS Computational Biology 3(10): e189. doi:10.1371/journal.pcbi.0030189, 2007.
31. N. Hanusse, D. J. Kavvadias, E. Kranakis, and D. Krizanc. Memoryless search algorithms in a network with faulty advice. TCS 402(2-3), 190-198, 2008.
32. R.D. Harkness and N.G. Maroudas. Central place foraging by an ant (*Cataglyphis bicolor* Fab.): a model of searching. Animal Behavior 33(3) 916–928, 1985.
33. C.A. Janeway, P. Travers, M. Walport and M.J. Shlomchik Immunobiology: The Immune System in Health and Disease New Yoy: Garland Science, 2001.
34. M. Kao, J. H. Reif, and S. R. Tate. Searching in an Unknown Environment: An Optimal Randomized Algorithm for the Cow-Path Problem. *J. of Inf. Comput.*, (1996), 63–79.
35. A. Korman, J.S. Sereni, and L. Viennot. Toward More Localized Local Algorithms: Removing Assumptions Concerning Global Knowledge. In PODC 2011.
36. A. Korman and S. Kutten. Distributed Verification of Minimum Spanning Trees. *Distributed Computing (DC)* 20(4), 2007.
37. A. Korman, S. Kutten and D. Peleg. Proof Labeling Schemes. *Distributed Computing (DC)* 22(4), 2010.
38. J. Krebs. Optimal foraging, predation risk and territory defense. *Ardea*, (1980), 68, 83–90. Nederlandse Ornithlogische Unie.
39. A. López-Ortiz and G. Sweet. Parallel searching on a lattice. CCCG 2001: 125–128.
40. G. F. Orians, and N. E. Pearson. On the theory of central place foraging. Analysis of Ecological Systems (pp. 155–177), 1979.
41. P. Panaite and A. Pelc. Exploring unknown undirected graphs. *J. of Algorithms*, (1999), 33, 281–295.
42. M. M. Polycarpouy, Y. Yang, K. M. Passinoz. A Cooperative Search Framework for Distributed Agents. In *Intelligent Control.*, (2001), 1–6.
43. S.C. Pratt. Quorum sensing by encounter rates in the ant Temnothorax albipennis. Behavioral Ecology 16(2): 488-496, 2005.
44. F. Rieke, D. Warland and W. Bialek. Coding efficiency and information rates in sensory neurons. Europhysics Letters 22(2): 15–156, 1993.

45. O. Reingold. Undirected connectivity in log-space. *J. ACM* 55(4): (2008).
46. A.M. Reynolds. Cooperative random Lévy flight searches and the flight patterns of honeybees. Physics Letters A 354 (2006) 384–388.
47. A.M. Reynolds. Optimal random Lévy-loop searching: New insights into the searching behaviours of central-place foragers. European Physics Letters 82(2) 20001 (2008).
48. S. Sommer and R. Wehner. The ant's estimation of distance travelled : experiments with desert ants, *Cataglyphis fortis*. J. of Comparative Physiology A 190(1) 1–6. 2004.
49. M. V. Srinivasan, S. Zhang, M. Altwein and J. Tautz. Honeybee Navigation: Nature and Calibration of the Odometer. Science 287: 851-853 (2000).
50. M.G. Surette, M.B. Miller and B.L. Bassler BL. Quorum sensing in Escherichia coli, Salmonella typhimurium, and Vibrio harveyi: a new family of genes responsible for autoinducer production. Proc. National Acadamy of Science 96:1639–1644, 1999.
51. C.D. Town, J.D. Gross and R.R. Kay. Cell differentiation without morphogenesis in Dictyostelium discoideum. Nature 262: 717–719, 1976.
52. R. Wehner, C. Meier and C. Zollikofer. The ontogeny of foraging behaviour in desert ants, *Cataglyphis bicolor*. Ecol. Entomol. 29, 240–250 (2004).
53. R. Wehner and M.Y. Srinivasan. Searching behaviour of desert ants, genus *Cataglyphis (Formicidae, Hymenoptera)* J. of Comparative Physiology 142(3) 315–338 (1981)
54. A. Zahavi. The function of pre-roost gatherings and communal roosts. Ibis 113: 106–109, 1971.