Towards More Realistic ANTS

Yuval Emek[†] Tobias Langner[§] David Stolz [§] Jara Uitto[§] Roger Wattenhofer[§]

[†] Technion, Israel [§] ETH Zürich, Switzerland

Abstract

In this paper, we summarize our results on a variant of the *Ants Nearby Treasure Search (ANTS)* problem, where n mobile agents, controlled by asynchronous randomized finite automata, search collaboratively for a treasure hidden by an adversary. We show that despite these restrictions, the treasure can be located in an optimal run-time of $\mathcal{O}(D + D^2/n)$. Moreover, we consider slight variations of the above model and showed that the minimum of agents required to solve the ANTS problem depends crucially on the computational capabilities of the agents as well as the timing parameters of the execution environment. Lastly, we present a protocol for a synchronous environment that allows the agents to locate the treasure in time $\mathcal{O}(D + D^2/n + Df)$ when f agents may crash at any time during the execution, while we allow f to be at most a constant fraction of n.

Contact Author Information

Tobias Langner
ETH Zurich
Gloriastrasse 35, ETZ G61.4
8092 Zurich
Switzerland

phone: +41 44 63 27730
email: tobias.langner@tik.ee.ethz.ch

1 Introduction

"They operate without any central control. Their collective behavior arises from local interactions." The last quote is arguably the mantra of distributed computing, however, in this case, "they" are not nodes in a distributed system; rather, this quote is taken from a biology paper that studies social insect colonies [13]. Understanding the behavior of insect colonies from a distributed computing perspective will hopefully prove to be a big step for both disciplines.

We consider the *Ants Nearby Treasure Search (ANTS)* problem where a colony of n ants (a.k.a. *agents*) whose nest is located at the origin of an infinite grid collaboratively search for an adversarially hidden treasure. An ant can move between neighboring grid cells and can communicate with the ants that share the same grid cell. However, the ant's navigation and communication capabilities are very limited since its actions are controlled by a randomized *finite state machine* (FSM) with a constant number of states.

The main theme of this research project is to study the ANTS problem under weaker assumptions than those considered in previous work. Our goal is to bring the problem closer to a "realistic" and "natural" setting by allowing very limited communication and restricting the computational power of the agents. We show that solving the ANTS problem is possible even under these weaker assumptions and when the agents are subject to adversarial conditions both with respect to their timing parameters and crash failures.

Results. As the first result in our model, we designed a distributed algorithm ensuring that the ants locate the treasure within $\mathcal{O}(D + D^2/n)$ time units w.h.p., where D denotes the distance between the origin and the treasure [5].[1] In this work, we assumed a fully asynchronous model and that in the beginning of the execution, the agents are indistinguishable.

Next, we studied computability aspects of the ANTS problem. We analyzed the minimum number of agents that are necessary to solve the problem [4]. We showed that a single randomized FSM cannot locate the treasure in finite expected time and that two deterministic ants cannot locate the treasure even if the model is synchronous (in the deterministic model, we assume that the agents can be distinguished from each other). On the positive side, we showed that three ants are sufficient if randomization is allowed or if the model is synchronous. Furthermore, we showed that four ants suffice in the asynchronous deterministic model.

In addition, we studied the effects of crash failures in the treasure search [10]. In essence, we implement an error checking mechanism that detects if an agent crashed and keeps track of the distance of the cells that have been searched successfully. In case of a failure, we restart the search from the closest distance to the origin that has not yet been searched successfully. We show that the treasure can be located in time $\mathcal{O}(D + D^2/n + Df)$, where f is the number of failures.

Related Work. Feinerman et al. [7, 8] introduce the aforementioned ANTS problem and study it, assuming that the agents are controlled by a Turing machine (with or without space bounds) and do not communicate with each other outside the origin. They show that if the n agents know a constant approximation of n, then they can find the treasure in time $\mathcal{O}(D + D^2/n)$ and observe a matching lower bound. Lenzen et al. studied the effects that bounding the memory of the agents and the range of available probabilities have on the runtime [11]. More generally, in graph exploration, the goal is for a single agent to visit all nodes in a given graph [1, 2, 3, 12, 14]. The line of work closest to ours is probably the one studying graph exploration by FSM-controlled agents [9]. Notice that in the case of an infinite grid, the expected time it takes for a random walk to reach any designated cell is infinite.

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

2 Model

Consider n mobile agents that explore $\mathbb{Z}^2$. In the beginning of the execution, all agents are positioned in a designated grid cell referred to as the *origin* (say, the cell with coordinates $(0,0) \in \mathbb{Z}^2$). The agents are controlled by an asynchronous randomized *finite state machine* (FSM) with a constant number of states. This means that the individual agent has a constant memory and thus, in general, can store neither coordinates in $\mathbb{Z}^2$ nor the number of agents. The agents in our model are allowed to communicate with each other. Specifically, an agent a positioned in cell $c \in \mathbb{Z}^2$ can communicate with all other agents positioned in cell c at the same time. This communication is quite limited though: agent a merely senses for each state q of the finite state machine, whether there exists at least one agent $a' \neq a$ in cell c whose current state is q. Notice that this communication scheme is a special case of the one-two-many communication scheme introduced in [6] with bounding parameter $b = 1$.

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

The agents operate in an asynchronous environment. Each agent's execution progresses in discrete (asynchronous) steps indexed by the non-negative integers and we denote the time at which agent a completed step $i > 0$ by $t_a(i) > 0$. Following the common practice, we assume that the time stamps $t_a(i)$ are determined by the policy ψ of an adversary that knows the protocol but is oblivious to its random bits, whereas the agents do not have any sense of time.

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

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

is the *transition function*. At time 0, all agents are in state s_0 and positioned in the origin. Suppose that at time $t_a(i)$, agent a is in state $q \in Q$ and positioned in cell $c \in \mathbb{Z}^2$. Then, the state $q' \in Q$ of a at time $t_a(i+1)$ and its corresponding position transition $\tau \in \{N,S,E,W,P\}$ are dictated based on the transition function δ by picking the pair $(q',\tau) \in \delta(q,Q_a)$, uniformly at random from $\delta(q,Q_a)$, where $Q_a \subseteq Q$ contains state $p \in Q$ if and only if there exists some (at least one) agent $a' \neq a$ such that a' is in state p and positioned in cell c at time $t_a(i)$. (Step i is deterministic if $|\delta(q,Q_a)| = 1$.) For simplicity, we assume that while the state subset Q_a (input to δ) is determined based on the status of cell c at time $t_a(i)$, the actual application of the transition function δ occurs instantaneously at the end of the step, i.e., agent a is considered to be in state q and positioned in cell c throughout the time interval $[t_a(i), t_a(i+1))$.

The goal of the agents is to locate an adversarially hidden *treasure*, i.e., to bring at least one agent to the cell in which the treasure is positioned. The distance of the treasure from the origin is denoted by D. As in [8], we measure the performance of a protocol in terms of its run-time, where the time is scaled so that $t_a(i+1) - t_a(i) \leq 1$ for every agent a and step $i \geq 0$. Although we express the run-time complexity in terms of the parameters n and D, neither of these two parameters is known to the agents (who cannot even store them in their very limited memory).

3 Selected Results

Our goal in this section is to present a selection of our results towards a deeper understanding of the ANTS problem. We have elected to present the two search strategies DiamondSearch and GeometricSearch as they play a central role in several contributions. Furthermore, we present an exemplary lower bound on the minimum number of agents required to solve the ANTS problem in a synchronous variant of the above model.

Diamond Search. The DiamondSearch strategy uses an *emission scheme*, which emits teams consisting of five agents from the origin one after the other. Due to space limitations, we will not explain how the emission works and refer the interested reader to the original publication [5].

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

The second stage of DiamondSearch is started when the last search team is emitted from the origin. At this point in time, $\Theta(n)$ cells are occupied by sleeping guides/explorers in all four cardinal directions. The last search team wakes up the innermost sleeping search team upon which it resumes its job and walks outwards to explore the next unexplored level in the same way as in the first stage. Each team recursively wakes up the search team of the next level until all sleeping teams have been woken up and resumed the search. A search in the second stage has one important difference in comparison to a search in the first stage: When an explorer meets a guide g during a search, instead of entering a sleep state, g moves outwards to the next unexplored level, hopping over all the other stationary guides on its way, and waits there for the next appearance of an explorer. When the explorer has finished its rectangle by reaching the north-guide again, it moves north (with the north-guide) to the first unexplored level and starts another search there. An illustration of DiamondSearch for a single search team is given in Figure 1.

It is to easy that DiamondSearch eventually explores every cell. Indeed, we showed that even in an asynchronous environment, the agents locate the treasure in an optimal runtime of $\mathcal{O}(D^2/n+D)$.

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

Initially, the agent moves one cell to the east. Then, it moves a geometrically distributed number of steps east following which it moves a geometrically distributed number of steps to the north. More precisely, with probability $1/2$ the agent moves further and otherwise stops walking in the current direction. This can be realized in our model by having two state transitions where one of them moves the agent further while the other one ends the walk. Either of the two transitions is chosen uniformly at random and a walk of geometrically distributed length is obtained. GeometricSearch guarantees that any cell within distance $D \leq \log(n)/2$ is visited by at least one agent w.h.p. [5].

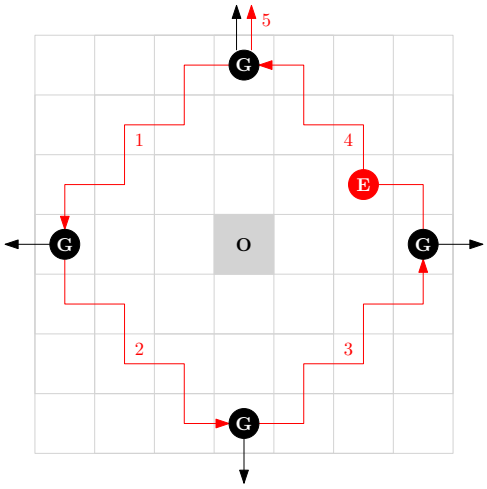

Figure 1: An explorer (E) starts a rectangle search in distance $d = 3$ at the north-guide, visits all guides (G) in distance d in a counter-clockwise fashion and ends at the north-guide ($1 - 4$). Whenever the explorer meets a guide, the guide moves outwards in its cardinal direction. When the explorer completes the search by arriving again at the north-guide, both agents walk outwards to the next level to be searched (5).

Lower Bounds. So far, we have considered a setting in which large numbers of agents are available and analyzed the speed-up that many agents can achieve compared to a constant number. On the other end of the spectrum, we also studied how many agents are minimally required to solve the ANTS problem when their computational power is varied slightly.

Consider the ANTS problem in a synchronous environment ($t_a(i) = i$ for all agents a and steps i). We show that this variant of the ANTS *cannot* be solved by two agents controlled by deterministic finite automata [4]. A crucial step towards this result is to observe that any protocol that locates the treasure must exhibit an infinite number of meetings between the agents. For a protocol $\mathcal{P}$ controlling only two agents, we show that there must exist an infinite sequence of steps in which the two agents meet in the same state combination. Thus, between any two such steps, the agents execute the same set of transitions and movements and therefore can only explore cells in a band with finite width. Consequently, they cannot locate the treasure.

4 Ongoing Work

The two most straightforward open questions related to the ANTS problem with finite state machines are whether three asynchronous or two randomized ants are sufficient to find the treasure. Answering these two questions closes the gaps left open in [4] and thus, completes the picture of the computability aspects of treasure search of finite state machines. In addition, one may wonder if locating the treasure is indeed the "right" goal for agents whose navigation capabilities are so weak (cannot even store their own coordinates). Due to the nature of our problem, one may ask, for example, if our algorithms can be extended to ensure that, once the treasure has been located, the agents are able to find their way back to the origin. Furthermore, can we ensure that each agent will be able to return to the origin and does this requirement have an impact on the runtime?

One may argue that our algorithms do not really seem to resemble the way that real ants search for their food. Yet, for now, we do not claim that our results explain any natural phenomenon. An interesting problem is how to modify our models and algorithms such that the behavior of the agents comes closer to the natural foraging behavior of real ants.

References

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

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

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

[4] Emek, Y., Langner, T., Stolz, D., Uitto, J., Wattenhofer, R.: How Many Ants Does it Take to Find the Food? In: 21th International Colloquium on Structural Information and Communication Complexity (SIROCCO). (2014)

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

[6] Emek, Y., Wattenhofer, R.: Stone Age Distributed Computing. In: Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC). (2013)

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

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

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

[10] Langner, T., Stolz, D., Uitto, J., Wattenhofer, R.: Fault-Tolerant ANTS. In: Proceedings of the 28th International Symposium on Distributed Computing (DISC). (2014) To appear.

[11] 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 33rd Symposium on Principles of Distributed Computing (PODC). (2014)

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

[13] Prabhakar, B., Dektar, K.N., Gordon, D.M.: The Regulation of Ant Colony Foraging Activity Without Spatial Information. PLoS Computational Biology **8**(8) (August 2012)

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