ANTS on a Plane*

Abhinav Aggarwal and Jared Saia

Dept. of Computer Science,
University of New Mexico, New Mexico, USA
{abhiag,saia.cs}@unm.edu

Abstract. In the ANTS (Ants Nearby Treasure Search) problem, multiple searchers, starting from a central location, search for a treasure. The searchers cannot communicate and have few bits of initial knowledge, called *advice*, when they begin the search. In this paper, we initiate the study of ANTS in the geometric plane.

Our main result is an algorithm, GoldenFA, that tolerates arbitrarily many crash failures caused by an adaptive adversary, and requires no bits of advice. GoldenFA takes $O\left(\left(L + \frac{L^2(t+1)}{ND}\right)\log L\right)$ expected time to find the shape, for a shape of diameter D, at distance L from the central location, with N searchers, $t < N$ of which suffer adversarial crash-failures.

We complement our algorithm with a lower bound, showing that it is within logarithmic factors of optimal. Additionally, we empirically test GoldenFA, and a related heuristic, and find that the heuristic is consistently faster than the state-of-the-art. Our algorithms and analysis make critical use of the Golden Ratio.

Keywords: Golden Ratio, Reliability, Computational Geometry, Natural Algorithms

1 Introduction

How can multiple simple searchers best find a target? Feinerman, Korman and others formalized this question by defining the ANTS (Ants Nearby Treasure Search) problem, where many searchers, all starting at a central location, seek a hidden target [10, 11, 9]. In this paper, we extend results on the ANTS problem in two key directions. Our first extension is to consider search on a 2-dimensional plane, rather than on a grid graph. This has two advantages for applications involving geometric search.[1] First, it allows us to more easily design search algorithms for targets of different sizes and shapes. Second, it avoids the problem of choosing the correct granularity for the grid graph. In particular, if the granularity is too low, then the target may not overlap any node. But if the granularity is too high, it places a high computational burden on the searchers.

* With apologies to the cast and crew of the Hollywood classic *Snakes on a Plane*. This work is supported by the National Science Foundation grant CNS 1816250.

[1] Our own motivating application is drones searching for gas plumes [30, 1].

Our second extension is ensuring provable robustness to adversarial failures, without requiring communication among searchers. Importantly, our algorithm can tolerate all but 1 searcher crashing, and the efficiency of our algorithm decreases only linearly with the actual number of faults, even when that number is not known in advance.

Our Model. N searchers start at a central location, called the *nest*. We define a *treasure* to be a convex shape with ratio of diameter to width equal to a fixed constant. Recall that the *diameter* of a convex shape is the largest distance between two parallel lines that are both tangent to the boundary of the shape, and that the *width* is the smallest such distance.[2]

A treasure of diameter D is placed adversarially at a distance L from the nest, where L is measured from the nest to the geometric center of the treasure. The searchers are synchronous in the sense that they all move at the same speed, and that local computation is instantaneous. The searchers cannot communicate with each other, and have zero bits of initial knowledge, including no knowledge of L or D. We measure the time it takes for some searcher to first locate the treasure. We refer to this as the *search time* of our algorithm. Our failure model is based on an adaptive adversary considered in [23]. In particular, an omniscient adversary chooses $t < N$ searchers that suffer crash failures at times chosen by the adversary.

We assume that every searcher has the ability to turn at angles of both π and $2\pi/\phi$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the Golden Ratio. When turning at an angle of α, let $\beta = 2\pi - \alpha$, be the remaining angle in the circle. Then to turn at an angle of π, requires that the searcher has the ability to turn until $\alpha = \beta$. To turn at an angle of $2\pi/\phi$, requires that the searcher have the ability to turn until $\frac{2\pi}{\alpha} = \frac{\alpha}{\beta}$.

1.1 Our Results

Our upper bound, summarized in the theorem below, considers N searchers looking for a treasure of diameter D at distance L, with $t < N$ crash failures.

Theorem 1. *There exists an algorithm,* GoldenFA*, that in the presence of up to $t < N$ crash failures, is able to locate a treasure of unknown diameter D, placed adversarially at an unknown distance L from the nest, in expected search time $O\left(\left(L + \frac{L^2(t+1)}{ND}\right)\log L\right)$.*

Additionally, GoldenFA requires zero bits of initial knowledge, called *advice*; it is uniform in that the searchers know nothing about N, and have no unique identifiers.

We prove lower-bounds, showing that the expected run time of GoldenFA is within logarithmic factors of optimal among a class of spoke-based algorithms. A *spoke-based* algorithm is one where the searchers only search along line segments, where each line-segment has an end-point in the nest, the central location where the searchers all start. See Section 6 for details.

[2] For example, a treasure can be a circle, regular polygon, or rectangle with constant aspect ratio.

Algorithm	Advice (bits)	Robustness	Runtime
F&K (advice)	$O(\log \log N)$	Not Robust	$O\left(L + \frac{L^2}{N}\right)$ for $D = \Theta(1)$
F&K (no advice)	0	Not Robust	$O\left(\left(L + \frac{L^2}{N}\right) \log^{1+\varepsilon} N\right)$ for fixed $\varepsilon > 0$ and $D = \Theta(1)$
GoldenFA	0	$t < N$	$O\left(\left(L + \frac{L^2(t+1)}{ND}\right) \log L\right)$

Table 1. A comparison of GoldenFA and the algorithms by Feinerman and Korman [10] (abbreviated as F&K). While the latter are not provably robust against adversarial crash failures, GoldenFA can efficiently handle all but one searchers to fail, even when these failures are scheduled by an adaptive adversary.

Our algorithm makes use of the Golden Ratio, both to ensure robustness and to ensure good coverage during the search. To the best of our knowledge, our algorithm is the first for the ANTS problem that makes use of this value.

1.2 Novelty and Technical Challenges

Our upper bound makes critical use of the Golden Ratio, and the difficultly to approximate it rationally. In particular, we can write any number as a (possibly infinite) continued fraction [18] of the form $x_1 + \frac{1}{x_2 + \frac{1}{x_3 + \cdots}}$, where the x_i values are all integers for $i \geq 1$. The degree to which the original number is well-approximated by a finite continued fraction depends on how large the x_i values are. For example, if x_2 is large, then the absolute difference between x_1 and the original number is small; if x_3 is large, then the absolute difference between $x_1 + 1/x_2$ and the original number is small, and so forth.

When $x_i = 1$ for all $i \geq 1$, we obtain an irrational number that is most difficult to approximate. To find this most difficult to approximate irrational number, we set $y = 1 + \frac{1}{y}$, and solve the resulting quadratic equation to obtain a solution $y = \frac{1+\sqrt{5}}{2}$, which is the celebrated Golden Ratio ϕ.

Using ϕ to spread-out spokes. In our algorithm, searchers proceed from the nest in line segments that we call *spokes*. Each new spoke is oriented at arc length ϕ, along the unit circle, from the previous one. The fact that ϕ is difficult to approximate with a rational number has useful implications in ensuring the angles between spokes are "well-spread". For example, if we start at the point 0 on a unit circle, and iteratively add points by moving clockwise by arc distance ϕ, then we will end up with near uniform distance between points (See Lemma 3 and [20, 29]). In particular, if x spokes are added this way, then the maximum arc length on a unit circle between neighboring spokes is $O(1/x)$ by the Three Gap Theorem (see Lemma 3). This allows us to locate the treasure efficiently, when D is unknown. Interestingly, this has connections to how plants add leaves as they grow. In particular, if the next leaf is added by moving arc length ϕ

along a unit circle, this ensures that leaves are well-spread, which increases their exposure to sunlight [27].

Using ϕ to handle failures. In our algorithm, each searcher creates the first spoke at a random heading and then iteratively proceeds to the next spoke by moving an arc distance ϕ along the unit circle (see Figure 1). Thus, even in the presence of $t = N - 1$ failures, the gaps between the spokes generated by the single remaining searcher decrease linearly and the treasure is found. This way, our algorithm eventually succeeds even when all but one searcher crashes.

Unknown L. Since L is unknown, we must carefully balance increasing spoke lengths and decreasing arc lengths between spokes over time. Simple doubling of spoke lengths over time is inefficient. Instead, our algorithm proceeds in epochs, where in epoch i, we search along spokes of length $2^0, 2^1, \ldots, 2^{i-1}$. In each epoch, we ensure that the amount of time spent searching along spokes of length 2^j is the same for all $0 \leq j < i$. We do this by having 2^{i-1} spokes of length 2^0, 2^{i-2} spokes of length 2^1, and so on up to 2^0 spokes of length 2^{i-1} (see Figure 2). Additionally, the angles between these spokes is determined using the Golden Ratio so that the angular gaps decrease linearly with the number of spokes.

1.3 Paper Organization

The rest of the paper is organized as follows. We discuss related work in Section 2 and some technical preliminaries in Section 3. We describe GoldenFA in Section 4 and analyze it in Section 5. We then give our lower bounds in Section 6. We provide empirical results on GoldenFA, comparing it with existing work, in Section 7. Finally, we conclude and discuss areas for future work in Section 8.

2 Related Work

Search is a fundamental problem in biology, where survival depends on search for mates, prey and other resources. It is also a common problem in robotics and mobile computing. Collective search, where multiple searchers must coordinate, is a key problem in computer science, robotics and in social insects. Ant- and bee-inspired algorithms have been particularly influential in swarm robotics research [28, 22, 16].

ANTS. Feinerman, Korman et al. [11, 9, 10] introduced the ANTS problem where multiple searchers starting from the same central location search for a treasure. Searchers are simple in that they cannot communicate and have few bits of initial knowledge, called advice, when they first leave the nest. Research on this problem now extends in multiple directions including: tradeoffs between computational resources and knowledge of searchers and the search time [9, 24, 7]; tradeoffs between communication and search time [25, 23, 3]; fault-tolerance [23]; handling asynchronous searchers [25, 8]; and game theoretic analysis of rational searchers [4]. As stated previously, our model is equivalent to that of [11, 9, 10], except that we search for a convex treasure in the infinite plane, rather than

a single vertex on an infinite grid. We note that the paper [11], while alluding to search on the plane, actually performs search on a two dimensional grid, by assuming each agent has a "bounded field of view of say ε" (Section 2 of [11]).

There are two potential benefits to avoiding this type of discretization. First, in some search applications, such as gas plume detection [30], there may be no clear analogue to a bounded "field of view". In this case, choosing ε too large risks missing the treasure, but choosing ε too small increases computational load on the searchers, since coordinate storage space seems to grow as $\Omega(\log(1/\varepsilon))$. Second, searching in the geometric plane more naturally allows for consideration of different shapes and sizes for the treasure. In many search applications, this seems important since targets are likely to be large or to be co-located, in both biological [13, 2] and engineering systems [14, 30].

Golden Ratio. Our algorithms make critical use of the celebrated Golden Ratio. This ratio is the limit of the ratio of consecutive numbers in the Fibonacci sequence. Fibonacci generated the sequence as an idealized model of a reproducing rabbit population assuming overlapping generations [6]. It was documented in India many centuries earlier, and has been observed in numerous biological systems including the arrangement of pine cones, unfurling of fern leaves, and the arrangement of sunflower seeds that optimally fills the circular area of the flower [27]. The Golden Ratio and Fibonacci numbers have been used in computer science for various applications like obtaining optimal schedules for security games [17], Fibonacci hashing [20], bandwidth sharing [15], data structures [12] and game theoretic models for blocking-resistant communication [19]. See [26] for a fascinating discussion of the history and applications of the Golden Ratio.

Crash Faults. To the best of our knowledge, work by Langner et al. [23] is the only other result that tolerates adversarial crash failures for a problem similar to ANTS. However, their model significantly deviates from ANTS in that they allow communication. In particular, constant-sized messages can be exchanged between searchers when they are both at the same location. Additionally, their searchers are much more restricted than ours in that they are modeled by finite-state automata. They describe an algorithm that locates a single target in $O\left(L + L^2/N + Lt\right)$ time, while tolerating $t \leq cN$ crash failures for some constant $c < 1$. In contrast, our algorithm can handle any $t < N - 1$, and does not require communication.

3 Technical Preliminaries

Let $\phi = (1 + \sqrt{5})/2$ denote the Golden ratio. For $m \geq 1$, let F_m denote the m^{th} Fibonacci number, so that $F_1 = F_2 = 1$ and $F_m = F_{m-1} + F_{m-2}$ for all $m \geq 3$. Given integer n, let $m(n)$ denote the index of the largest Fibonacci number not greater than n.

Lemma 1. *For all $x \geq 1$, the following properties hold:*

1. $\lfloor \log_\phi x + 1 \rfloor \leq m(x) \leq \lceil \log_\phi x + 2 \rceil$.

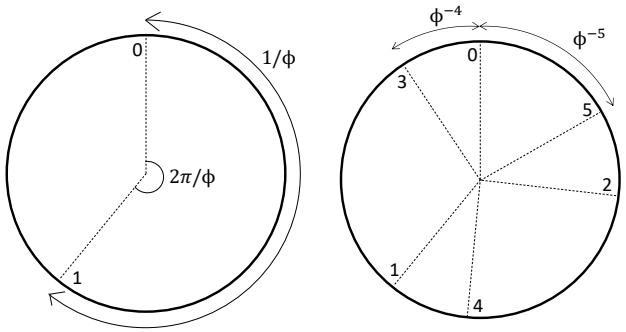

Fig. 1. A schematic of the gaps induced by points placed on the unit circle, following Lemma 3. When the arc distance between successive points is $\phi \equiv \phi^{-1} \mod 1$, then the gaps decrease as the number of points increase.

2. $\frac{1}{\phi^3 x} \leq \phi^{-m(x)} \leq \frac{1}{x}$.

Proof. For (1), using the fact that $F_r \leq \phi^{r-1}$ for all r, it holds that $F_{\lfloor \log_\phi x+1 \rfloor} \leq \phi^{\lfloor \log_\phi x+1 \rfloor - 1} \leq x$. Similarly, since $F_r \geq \phi^{r-2}$ for all r, it holds that $F_{\lceil \log_\phi x+2 \rceil} \geq \phi^{\lceil \log_\phi x+2 \rceil - 2} \geq x$. For (2), using the result obtained in (1), we obtain $\phi^{-m(x)} \leq \phi^{-\lfloor \log_\phi x+1 \rfloor} \leq \frac{1}{x}$. Similarly, $\phi^{-m(x)} \geq \phi^{-\lceil \log_\phi x+2 \rceil} \geq \frac{1}{\phi^3 x}$. □

We define a *unit circle* as a circle around the nest with circumference one.

Lemma 2. *Let the treasure be oriented so that its diameter is perpendicular to the spoke ending at the diameter midpoint. Let α be the arc length on the unit circle made by the two spokes that are tangent to the diameter. Then,*

1. $\alpha = \frac{1}{\pi} \sin^{-1}\left(\frac{D}{2L}\right)$*; and*
2. $\alpha \geq \frac{D}{2\pi L}$.

Proof. Part (1) follows by definition. Part (2) follows from the Maclaurin expansion [21] of $\sin^{-1} x$, from which it follows that $\sin^{-1} x \geq x$. □

Our analysis makes use of the following lemma regarding the Three Gap Theorem by Swierczkowski [29] (also known as the Steinhaus Conjecture) for Golden-ratio based gaps between successive points on the circumference of a unit circle (see Figure 1). In this lemma, the set of points on the unit circle is equivalent to the set of points generated by our algorithm. This holds since $\phi^{-1} \equiv \phi \mod 1$, because $\phi^{-1} = \phi - 1$. The last sentence of the lemma follows immediately from Lemma 1(2).

Lemma 3 (Restatement of Corollary 2 from [29]). *Let C be a circle of circumference 1 and p_0 be a fixed starting point on C. For $k \geq 0$, let p_k be the point which makes an arc of length $k\phi$ from p_0, measured clockwise. Let $n \geq 1$*

Algorithm 1: The GoldenFA Algorithm.

```
   /* Each searcher independently performs the following steps.     */
1  i ← 1;
2  while treasure not found do
3  |  direction ← uniformly random heading on the unit circle;
4  |  for j ∈ {0, ..., i − 1} do
5  |  |  Traverse 2^{i−j} spokes of length 2^j. The first spoke is at heading
   |  |   direction. Each subsequent spoke has heading that increases
   |  |   clockwise by arc distance of φ, along the unit circle, from the heading
   |  |   of the previous spoke.;
6  |  end
7  |  i ← i + 1;
8  end
```

and F_m be the largest Fibonacci number no more than n. Then, the set of points $P_n = \{p_0, p_1, \ldots, p_n\}$ partition C into disjoint arcs, each of which has length ϕ^{-m}, ϕ^{-m+1} or ϕ^{-m+2}. In particular, this implies that every disjoint arc has length between $\frac{1}{\phi^3 n}$ and $\frac{\phi^2}{n}$.

In the rest of the paper, we will assume circular treasures but our results hold for all convex shapes where the ratio of the diameter to the width is a fixed constant. We assume that L is the distance from the nest to the center of the circular treasure.

Our algorithm is designed to search in the real plane, $\mathbb{R}^2$. We note, however, that it can be adapted to search in the infinite two-dimensional grid as follows. For every spoke generated by our algorithm, create a walk on the grid that visits every edge incident to every face in the grid that is intersected by the spoke. This ensures that we will find any treasure that overlaps a grid vertex. Additionally, it increases total search time by at most a constant factor.

4 GoldenFA

Algorithm 1 describes our main algorithm, GoldenFA. The algorithm proceeds in *epochs* numbered iteratively starting at $i = 1$. In epoch number, i, each searcher initially chooses a random initial heading *direction*. Then for all j, $1 \leq j \leq (i-1)$, the searcher traverses along 2^{i-j} spokes of length 2^j. Each spoke starts and ends at the nest. For each value of j, the first of these spokes is at heading *direction*, and each subsequent spoke has heading that increases clockwise along the unit circle at arc length of ϕ from the previous spoke. Thus in epoch i, a total of $\sum_{j=0}^{i-1} 2^{i-j} = 2^{i+1} - 1$ spokes are traversed. If the treasure is not found after these traversals, epoch i ends and epoch $i+1$ begins.

Figure 2 illustrates two epochs of GoldenFA when $N = 2$.

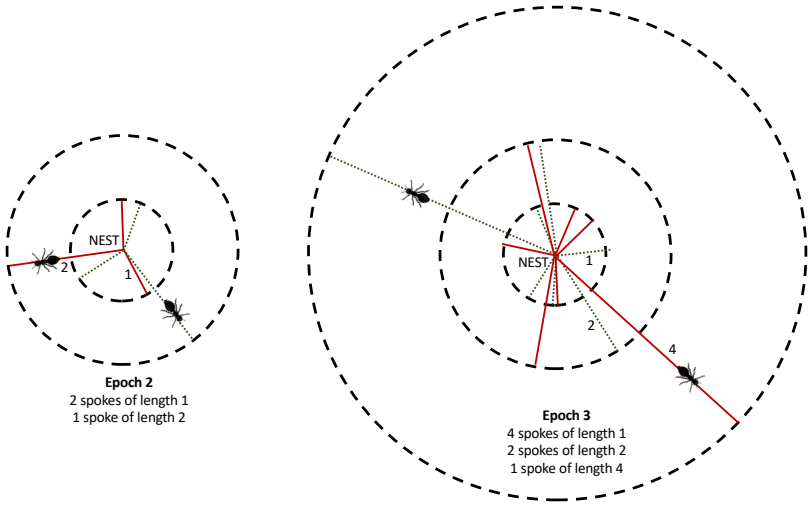

Fig. 2. A schematic of GoldenFA is shown for the spokes made by two searchers in epochs 2 and 3 (red solid lines for searcher 1 and green dotted lines for searcher 2). Both searchers choose a random initial heading at the beginning of every epoch.

5 Analysis

We next analyze GoldenFA and compute its runtime as a function of the unknown parameters: treasure diameter, its distance from the nest, the number of searchers and the number of crash failures. In the following, all log terms are base 2.

Lemma 4. *In epoch $i \geq \log L$, the probability that a single searcher finds the treasure is at least $\alpha 2^{i-(\log L)-5}$, where $\alpha \geq \frac{D}{2\pi L}$. In any epoch $i \geq \log L + \log \frac{\phi^2}{\alpha} + 1$, a searcher finds the treasure with probability 1.*

Proof. When $i \geq \log L$, there will be $2^{i-\lceil \log L \rceil}$ spokes of length at least L, where the first of these spokes has a uniformly random orientation, and the remainder are spread out at successive clockwise arc distances of ϕ. By Lemma 3, the maximum arc length between any neighboring pair of these $2^{i-\lceil \log L \rceil}$ spokes is $\phi^2 2^{-i+\lceil \log L \rceil}$. By Lemma 2, if any of these spokes intersect an arc of length α, where $\alpha \geq \frac{D}{2\pi L}$, then the searcher will find the treasure. Thus, for $i \geq \log L + \log \frac{\phi^2}{\alpha} + 1$, a searcher is guaranteed to find the treasure.

By Lemma 3, the minimum arc length between any neighboring pair of x spokes is $\frac{1}{\phi^3 x}$. Thus, when $x \leq \frac{1}{\alpha \phi^3}$, all spokes are arc distance at least α apart. If there are x such spokes of length at least L, the probability that one of these spokes intersects the treasure is $x\alpha$. To see this, imagine fixing the x spokes, and then letting the α length arc associated with the treasure move uniformly at random on the circumference of the unit circle. The total measure of locations

where the treasure may fall so that it intersects a spoke is then $x\alpha$. Hence, when $\log L \leq i \leq \lceil \log L \rceil + \log \frac{1}{\phi^3 \alpha}$, the probability that a single searcher finds the treasure is at least $\alpha 2^{i-\lceil \log L \rceil} \geq \alpha 2^{i-(\log L)-1}$.

Finally, note that for $\lceil \log L \rceil + \log \frac{1}{\phi^3 \alpha} < i < \log L + \log \frac{\phi^2}{\alpha} + 1$, the probability that a single searcher finds the treasure is at least $1/\phi^3$. Thus, in this range, the probability of finding the treasure is at least $\alpha \frac{1}{\phi^5} 2^{i-\lceil \log L \rceil} \geq \alpha 2^{i-(\lceil \log L \rceil + 4)} \geq \alpha 2^{i-(\log L)-5}$. □

Theorem 2. *In the presence of up to $t < N$ crash failures,* GoldenFA *takes an expected number of times steps that is $O\left(\left(L + \frac{L^2(t+1)}{ND}\right) \log L\right)$.*

Proof. First, we consider the case where $2(t+1) > N$. By Lemma 4, when $i \geq \log L + \log \frac{\phi^2}{\alpha} + 1$, all searchers will find the treasure. Thus, in this case, the total time of GoldenFA is no more than

$$\sum_{i=1}^{\log L + \log \frac{\phi^2}{\alpha} + 1} i 2^i = O\left(\frac{L\phi^2}{\alpha} \log \frac{L\phi^2}{\alpha}\right).$$

This is the claimed number of time steps when $t = \Theta(N)$, since $\alpha = \Theta(D/L)$.

Next, assume that $2(t+1) \leq N$. We first compute the expected number of searchers that find the treasure in each epoch, and then use this expectation to bound, for each epoch, the probability that the total number of searchers that find the treasure is no more than the total number of faults.

For any epoch i, let S_i be a random variable giving the number of searchers that find the treasure in epoch i. By Lemma 4, and linearity of expectation, we have that for $\log L \leq i \leq \log L + \log \frac{\phi^2}{\alpha} + 1$,

$$E(S_i) \geq N\alpha 2^{i-(\log L)-5},$$

where $\alpha \geq \frac{D}{2\pi L}$. Since each searcher finds the treasure independently, we can use Chernoff bounds on S_i (See [5], Exercise 1.1). These show that $Pr(S_i < (1/2)\mu_L) \leq e^{-\mu_L/8}$, where $\mu_L = N\alpha 2^{i-(\log L)-5}$ is a lower bound on the expected value. Let

$$i^* = (\log L) + 5 + \max\left(0, \log \frac{2(t+1)}{N\alpha}\right).$$

Then $E(S_i) \geq 2(t+1)$, when $i \geq i^*$ and $2(t+1) \leq N$. Thus, for $i \geq i^*$,

$$Pr(S_i < t+1) \leq e^{-N\alpha 2^{i-(\log L)-8}}.$$

This bound holds even for $i \geq \log L + \log \frac{\phi^2}{\alpha} + 1$, since for i in that range, $Pr(S_i < t+1) = 0$, since each searcher finds the treasure with probability 1.

Let X be a random variable giving the number of epochs until more than t searchers find the treasure. Note that $Pr(X \geq i) \leq Pr(S_{i-1} < t+1)$, where

$Pr(S_0 < t+1) = 1$. Then, we can bound the expected search time of our algorithm as follows.

$$\begin{aligned}\sum_{i\geq 1} i2^i Pr(X \geq i) &= \sum_{i\geq 1} i2^i Pr(S_{i-1} < t+1) \\ &\leq \sum_{1\leq i<i^*} i2^i + \sum_{i\geq i^*} i2^i e^{-N\alpha 2^{i-(\log L)-8}}\end{aligned}$$

Let S_1 be the value of the first sum. Note that:

$$\begin{aligned}S_1 &= \sum_{1\leq i<i^*} i2^i \\ &= O\left(L\log L + \left(\frac{L(t+1)}{\alpha N}\right)\log\left\lceil\frac{L(t+1)}{\alpha N}\right\rceil\right)\end{aligned}$$

Let S_2 be the value of the second sum. Note that:

$$\begin{aligned}S_2 &= \sum_{i\geq i^*} i2^i e^{-N\alpha 2^{i-(\log L)-8}} \\ &\leq 2i^*2^{i^*}\sum_{j\geq 1} j2^j e^{-N\alpha 2^{j+i^*-(\log L)-9}} \\ &\leq 2i^*2^{i^*}\sum_{j\geq 1}\exp\left(\ln j + j\ln 2 - 2^{j+i^*-\left(\log\frac{L}{N\alpha}\right)-9}\right)\end{aligned}$$

In the above, the second line holds by noting that for all $j \geq 1$ and $x \geq 1$, $(x+j)2^{x+j} \leq 2(x2^x)(j2^j)$, and letting $x = i^*$. Next, we bound the exponent:

$$\begin{aligned}\ln j + j\ln 2 - 2^{j+i^*-\left(\log\frac{L}{N\alpha}\right)-9} &\leq \ln j + j\ln 2 - 2^{j-3} \\ &\leq -j\end{aligned}$$

In the above, the first line holds since $i^* = (\log L) + 5 + \max\left(0, \log\frac{2(t+1)}{N\alpha}\right) \geq 5 + \log\frac{2L(t+1)}{N\alpha} \geq 6 + \log\frac{L}{N\alpha}$. The second line holds when $j \geq 7$, since then $\ln j + j\ln 2 - 2^{j-3} \leq -j$. Hence, the infinite summation is $O(1)$. Thus, we have that

$$\begin{aligned}S_2 &\leq 2i^*2^{i^*}\sum_{j\geq 1}\exp\left(\ln j + j\ln 2 - 2^{j+i^*-\left(\log\frac{L}{N\alpha}\right)-9}\right) \\ &= O\left(L\log L + \left(\frac{L(t+1)}{\alpha N}\right)\log\left\lceil\frac{L(t+1)}{\alpha N}\right\rceil\right)\end{aligned}$$

By Lemma 2, $\alpha \geq \frac{D}{2\pi L}$, so the total expected cost of GoldenFA is:

$$O\left(L\log L + \left(\frac{L^2(t+1)}{ND}\right)\log\left\lceil\frac{L^2(t+1)}{ND}\right\rceil\right).$$

Finally, note that $\log\left\lceil \frac{L^2(t+1)}{ND} \right\rceil = O(\log L + \log(t+1) - \log N - \log D) = O(\log L)$, since $N \geq t+1$.

Thus, we can simplify the above to:

$$O\left(\left(L + \frac{L^2(t+1)}{ND}\right)\log L\right).$$

Note that this is tight since if the log term on the right is less than $\log L$, it means that $\frac{L^2(t+1)}{ND} \leq L$, in which case the first summand dominates. □

6 Lower Bound for Spoke-Based Algorithms

We now prove a lower bound on the number of time steps that *any* spoke-based algorithm must take to locate the treasure in the presence of adversarial crash failures.

Theorem 3. *In the presence of up to $t < N$ crash failures, any spoke-based algorithm requires $\Omega\left(L + \left(\frac{L^2(t+1)}{ND}\right)\log L\right)$ time steps to locate the treasure.*

Proof. By Yao's lemma [31], the search time of the best randomized algorithm equals the search time of the best deterministic algorithm against a known randomized adversary. Thus, we compute the search time for the best deterministic algorithm against a known but randomized adversarial placement of the treasure.
Adversarial Strategy. Let x be a positive integer and y be an integer chosen uniformly at random in $[0, x]$. Let $L = 2^y$ and $L/D = 2^{x-y}$. Choose an integer z uniformly at random in $[0, L/D]$. Let the treasure be an ellipse with diameter D and an arbitrary small width. Place this ellipse so that its center is at distance L from the nest, in the direction from the nest that is oriented at arc distance $\frac{zD}{L}$ along the unit circle. Rotate the ellipse so that its diameter is perpendicular to the ray connecting the nest and the center of the ellipse.

Lower Bound Against This Strategy. Assume the algorithm knows the value x; the randomized adversarial strategy above; and t, the number of faults that will occur. The algorithm can be represented as a sequence, σ, of tuples. Each tuple corresponds to some searcher's first visit to a region that is a possible treasure location. In particular, tuple (ℓ, a) corresponds to a visit to any point in the ellipse with center at distance ℓ from the nest, and orientation that is arc length a along the unit circle centered at the nest. The tuples in σ are all sorted by time of visit to first point in the ellipse, with ties broken arbitrarily.

First, note that there is 1 unique tuple of length 2^x: $(2^x, 0)$; 2 unique tuples of length 2^{x-1}: $(2^{x-1}, 0)$ and $(2^{x-1}, 1/2)$; 4 unique tuples of length 2^{x-2}: $(2^{x-2}, 0)$, $(2^{x-2}, 1/4)$, $(2^{x-2}, 1/2)$, $(2^{x-2}, 3/4)$; and so forth. Next, observe that each unique tuple, (ℓ, a), appears in σ at least $t+1$ times. This is necessary since each possible ellipse must be visited by $t+1$ searchers in order for the algorithm to be robust to t adversarial faults. Finally, note that visiting any point on the ellipse corresponding to (ℓ, a) requires movement of $\Omega(\ell)$, no matter at what

tuple, (ℓ', a') the searcher visiting (ℓ, a) was previously at. To see this, first note that if $\ell = \ell'$, then the distance travelled between these two tuples is $\Omega(\ell)$ since there must be a trip to the nest between the tuples, because the algorithm is spoke-based. Second, if $\ell \neq \ell'$, draw two squares centered at the nest, one enclosing (ℓ, a), and the other enclosing (ℓ', a'). Then, note that the minimum distance between the squares is $\Omega(\ell)$.

The expected total distance travelled by all searchers in the algorithm is then given as follows. Select a tuple in σ uniformly at random, and sum up the lengths of all tuples preceding, and including, the selected tuple in σ. Let X be the random variable giving this sum. Note that X stochastically dominates the following random variable, X': Let σ' be a sequence where each tuple in σ of the type (ℓ, a) is expanded to ℓ copies. Select a tuple uniformly at random in σ' and let X' be the index of the selected tuple.

Note that $E(X')$ is half the length of σ', and that the length of $\sigma' = (t+1)x2^x$. Thus, $E(X) \geq E(X') \geq (1/2)(t+1)x2^x$. Finally, note that, since there are N searchers, the expected search time is at least the total distance travelled by all searchers divided by N. By linearity of expectation, the expected search time is thus at least $\frac{(t+1)x2^x}{2N}$. Since $2^x = L^2/D$, then $\frac{(t+1)x2^x}{2N} = \Omega\left(\frac{(t+1)L^2 \log L}{ND}\right)$. The lower bound is this value plus L, since no matter the values of D, N and t, the total search time is always at least L. □

7 Empirical Evaluation

We implement GoldenFA and algorithms from [10] to empirically evaluate how search time changes as we increase: the ratio of the diameter of treasure to distance to treasure (D/L); the number of searchers (N); and the fraction of random crash failures (t/N). We compare GoldenFA to algorithms from Feinerman and Korman in [10].

7.1 Setup

We implemented four algorithms. GoldenFA is our algorithm from Section 4. F&K-Advice is Algorithm 1 from [10]; it requires $O(\log \log N)$ bits of advice. F&K-NoAdvice is Algorithm 2 from [10] with $\epsilon = .01$; it requires zero bits of advice. Since the value for ϵ in Algorithm 2 is not specified in [10], we conducted experiments to determine that the setting $\epsilon = .01$ performs well empirically.

GoldenFA-Heuristic is the last algorithm. In this algorithm, for epoch $i \geq 1$, there are $\lceil c(1+\alpha)^i \rceil$ spokes of length $(1+\alpha)^i$, for parameters $c, \alpha > 0$. Similar to the GoldenFA, each spoke in this is at arc distance equal to the Golden Ratio from the previous. We set $c = 1.9$ and $\alpha = 7$, since they perform well empirically.

In all of our experiments the treasure is a circle with diameter D. For each data point plotted, 150 trials were run and the average search time was plotted. The location of the treasure was kept fixed throughout all trials. The search time reported is time steps, where one time step is the amount of time it takes

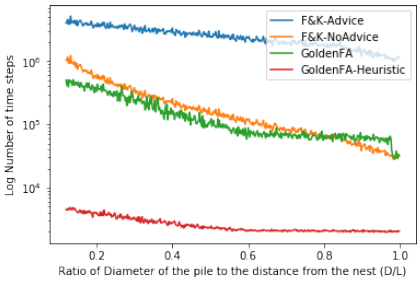

Fig. 3. Search time versus D/L; $L = 500$, $N = 1$, and D is varied.

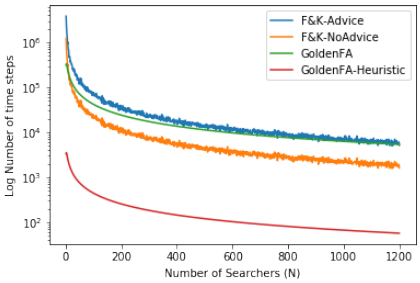

Fig. 4. Search time versus N; $L = 500$ and $D = 4$.

a searcher to travel a distance of 1. All algorithms were implemented in Python 3.6, and all experiments were run on a Macbook Pro with 2.6 GHz Intel Core i7 processor and 16GB RAM.

7.2 Results

Our results show how search time changes as we vary three different values. In particular, we include three plots giving results of experiments based on varying (1) the ratio D/L, where D is the treasure diameter and L the distance from the nest to the center of the treasure; (2) the number of searchers N; (3) and the fraction of faults t/N. In each plot, search time is the independent variable, and it is plotted on a logarithmic scale.

Search Time versus D/L. Our first experiment tracks search time as the ratio D/L increases. The value of L is fixed at 500, and D increases from 1 to 500.

Figure 3 shows how search time decreases as D/L increases from .1 to 1. As the plot shows, search time decreases for all algorithms. GoldenFA-Heuristic consistently has the best search time across values tested, with performance that is always between 1 and 2 orders of magnitude better than all other algorithms, when D/L is greater than about .15. Next, in performance, are GoldenFA and F&K-NoAdvice. Initially F&K-NoAdvice has worse search time than GoldenFA, but as D/L increases, they both trend towards roughly similar performance. Last, in the plot is F&K-Advice, which does not decrease nearly as much as the other algorithms as D/L increases.

It is surprising that F&K-NoAdvice performs better than F&K-Advice as D/L increases. We conjecture this holds because (1) F&K-NoAdvice has an algorithmic parameter (ϵ), while F&K-Advice has none; and (2) we optimized this parameter based on empirical feedback.

Search Time versus N. Our second experiment tracks search time versus the number of searchers, N. Figure 4 shows the outcome when $L = 500$, $D = 4$, and N varies from 1 to 200.

In this plot, search time of all algorithms decreases with N. GoldenFA-Heuristic performs about 2 orders of magnitude better than any other algorithm, for all values of N tested. Next comes F&K-NoAdvice, which performs up to a factor of about 5 better than the remaining algorithms. Finally, GoldenFA and F&K-Advice are last, with performance roughly equal for large N.

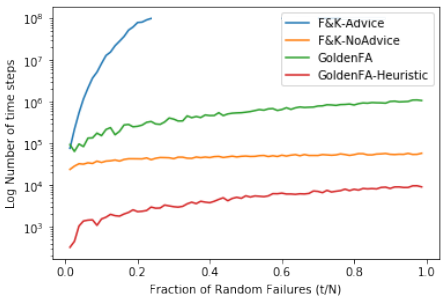

Fig. 5. Search time versus the fraction of failures (t/N); $L = 500, D = 4$ and $N = 100$.

Search Time versus t/N. Our last experiment tracks search time as the ratio t/N increases, where t is the number of crash failures and N is the number of searchers. In these experiments, we hold the following values fixed: $L = 500$, $D = 4$, $N = 100$; and we vary t from 0 to 99. For each value of t, a random subset of t searchers are removed after the first 100 time steps of the algorithm. To prevent any algorithm from running forever, a hard timeout was set at 10^8 time steps.

The results are given in Figure 5. Again GoldenFA-Heuristic has fastest search time over the entire range of values tested, with performance a bit less than an order of magnitude better than then next fastest algorithm, F&K-NoAdvice. F&K-NoAdvice has search times which increase slowly as t/N increases. GoldenFA comes next with a search time that increases more rapidly with t/N. Finally, F&K-Advice comes last, with search time increasing rapidly with t/N until it times out when t/N is roughly about .20. Our theoretical analysis suggests that search time for GoldenFA would increase roughly linearly with t/N. Results from this experiment suggest this is the case with slope of approximately 10 for search time as a function of t/N.

8 Conclusion and Future Work

We have described an algorithm, GoldenFA that solves the ANTS problem by finding a treasure that is a convex shape with any diameter D, even in the presence of $t < N$ crash failures. We have proven that our algorithm takes $O\left(\left(L + \frac{L^2(t+1)}{ND}\right)\log L\right)$ expected search time, where L is the distance from the nest to the treasure and N is the number of searchers. Additionally, we have proven a near-matching lower bound on search time for a class of "spoke-algorithms", which search only via line segments emanating from the nest. Our

algorithm uses the Golden Ratio to spread out search spokes uniformly, even in the presence of many crash failures.

Several interesting problems remain including the following. Can we develop a non-spoke-based algorithm that removes the logarithmic terms in our search time, but is still robust to failures and does not require advice? Fibonacci spirals are quite commonly used in nature for space-filling applications, so they may be useful for this open problem.

Another interesting open problem is to extend our results for multiple treasures with different shapes and orientations. It is possible for a treasure to have a large L but its orientation is such that the nearest point to the nest is only $\Theta(1)$ units away. This treasure can be located in $\Theta(1)$ time steps by searching along a spiral around the nest. However, when rotated, this treasure can be oriented in a way that now requires $O(L^2/D)$ time steps.

References

1. Barchyn, T.E., Hugenholtz, C.H., Fox, T.A.: Plume detection modeling of a drone-based natural gas leak detection system. Elementa Science of the Anthropocene **7**(1) (2019)
2. Beverly, B.D., McLendon, H., Nacu, S., Holmes, S., Gordon, D.M.: How site fidelity leads to individual differences in the foraging activity of harvester ants. Behavioral Ecology **20**(3), 633–638 (2009)
3. Boczkowski, L., Natale, E., Feinerman, O., Korman, A.: Limits on reliable information flows through stochastic populations. PLoS Computational Biology **14**(6), e1006195 (2018)
4. Collet, S., Korman, A.: Intense competition can drive selfish explorers to optimize coverage. In: Symposium on Parallelism in Algorithms and Architectures (SPAA) (2018)
5. Dubhashi, D.P., Panconesi, A.: Concentration of measure for the analysis of randomized algorithms. Cambridge University Press (2009)
6. Dunlap, R.A.: The Golden Ratio and Fibonacci numbers. World Scientific (1997)
7. Emek, Y., Langner, T., Stolz, D., Uitto, J., Wattenhofer, R.: How many ants does it take to find the food? Theoretical Computer Science **608**, 255–267 (2015)
8. Emek, Y., Langner, T., Uitto, J., Wattenhofer, R.: Solving the ANTS problem with asynchronous finite state machines. In: International Colloquium on Automata, Languages, and Programming (ICALP). pp. 471–482. Springer (2014)
9. Feinerman, O., Korman, A.: Memory lower bounds for randomized collaborative search and implications for biology. In: International Symposium on Distributed Computing (DISC). pp. 61–75. Springer (2012)
10. Feinerman, O., Korman, A.: The ANTS problem. Distributed Computing **30**(3), 149–168 (2017)
11. Feinerman, O., Korman, A., Lotker, Z., Sereni, J.S.: Collaborative search on the plane without communication. In: Proceedings of the 2012 ACM symposium on Principles of distributed computing (PODC). pp. 77–86. ACM (2012)
12. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM) **34**(3), 596–615 (1987)
13. Gordon, D.M.: Ants at work: how an insect society is organized. Simon and Schuster (1999)

14. Hecker, J.P., Carmichael, J.C., Moses, M.E.: Exploiting clusters for complete resource collection in biologically-inspired robot swarms. In: International Conference on Intelligent Robots and Systems IROS. pp. 434–440 (2015)
15. Itai, A., Rosberg, Z.: A Golden Ratio control policy for a multiple-access channel. IEEE Transactions on Automatic Control **29**(8), 712–718 (1984)
16. Karaboga, D., Akay, B.: A survey: algorithms simulating bee swarm intelligence. Artificial intelligence review **31**(1-4), 61–85 (2009)
17. Kempe, D., Schulman, L.J., Tamuz, O.: Quasi-regular sequences and optimal schedules for security games. In: ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 1625–1644. Society for Industrial and Applied Mathematics (2018)
18. Khinchin, A.I.: Continued fractions, vol. 525. P. Noordhoff (1963)
19. King, V., Saia, J., Young, M.: Conflict on a communication channel. In: Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing. pp. 277–286. ACM (2011)
20. Knuth, D.E.: The art of computer programming, vol. 3: Searching and sorting. Reading MA: Addison-Wisley (1973)
21. Kreyszig, E.: Advanced engineering mathematics. John Wiley & Sons, Inc. 9th edition (2008)
22. Krieger, M.J., Billeter, J.B., Keller, L.: Ant-like task allocation and recruitment in cooperative robots. Nature **406**(6799), 992 (2000)
23. Langner, T., Uitto, J., Stolz, D., Wattenhofer, R.: Fault-tolerant ANTS. In: International Symposium on Distributed Computing (DISC). pp. 31–45. Springer (2014)
24. 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 2014 ACM symposium on Principles of distributed computing (PODC). pp. 252–261. ACM (2014)
25. Lenzen, C., Radeva, T.: The power of pheromones in ant foraging. In: Workshop on Biological Distributed Algorithms (BDA) (2013)
26. Livio, M.: The golden ratio: The story of phi, the world's most astonishing number. Broadway Books (2008)
27. Naylor, M.: Golden, $\sqrt{2}$, and π Flowers: A Spiral Story. Mathematics Magazine **75**(3), 163–172 (2002)
28. Şahin, E.: Swarm robotics: From sources of inspiration to domains of application. In: International Workshop on Swarm Robotics. pp. 10–20. Springer (2004)
29. Świerczkowski, S.: On successive settings of an arc on the circumference of a circle. Fundamenta Mathematicae **46**, 187–189 (1958)
30. Williams, S.C.P.: Studying volcanic eruptions with aerial drones. Proceedings of the National Academy of Sciences **110**(27), 10881–10881 (2013)
31. Yao, A.C.C.: Probabilistic computations: Toward a unified measure of complexity. In: Symposium on Foundations of Computer Science (FOCS). pp. 222–227. IEEE (1977)