Search on a Line by Byzantine Robots[*]

Jurek Czyzowicz[1], Konstantinos Georgiou[2], Evangelos Kranakis[3], Danny Krizanc[4], Lata Narayanan[5], Jaroslav Opatrny[6], and Sunil Shende[7]

1 **Département d'informatique, Université du Québec en Outaouais, Gatineau, QC, Canada**
Jurek.Czyzowicz@uqo.ca

2 **Department of Mathematics, Ryerson University, Toronto, Canada**
konstantinos@ryerson.ca

3 **School of Computer Science, Carleton University, Ottawa, Canada**
kranakis@scs.carleton.ca

4 **Department of Mathematics and Computer Science, Wesleyan University, Middletown CT, USA**
dkrizanc@wesleyan.edu

5 **Department of Computer Science and Software Engineering, Concordia University, Montreal, QC, Canada**
lata@cs.concordia.ca

6 **Department of Computer Science and Software Engineering, Concordia University, Montreal, QC, Canada**
opatrny@cs.concordia.ca

7 **Department of Computer Science, Rutgers University, Camden, USA**
sunil.shende@rutgers.edu

Abstract

We consider the problem of fault-tolerant parallel search on an infinite line by n robots. Starting from the origin, the robots are required to find a target at an unknown location. The robots can move with maximum speed 1 and can communicate in wireless mode among themselves. However, among the n robots, there are f robots that exhibit *byzantine faults*. A faulty robot can fail to report the target even after reaching it, or it can make malicious claims about having found the target when in fact it has not. Given the presence of such faulty robots, the search for the target can only be concluded when the non-faulty robots have sufficient verification that the target has been found. We aim to design algorithms that minimize the value of $S_d(n, f)$, the time to find a target at a distance d from the origin by n robots among which f are faulty. We give several different algorithms whose running time depends on the ratio f/n, the density of faulty robots, and also prove lower bounds. Our algorithms are optimal for some densities of faulty robots.

1998 ACM Subject Classification F.1.2 Modes of Computation, Parallelism and concurrency, B.8 Performance and Reliability, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Cow path problem, Parallel search, Mobile robots, Wireless communication, Byzantine faults

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.27

[*] A full version of the paper is available at http://arxiv.org/abs/1611.08209.

1 Introduction

Searching on a line (also known as a *single-lane cow-path* or a *linear search*) problem is concerned with a robot looking for a target placed at an unknown location on an infinite line; the robot moves with uniform (constant) speed and can change direction (without any loss in time) along this line. The ultimate goal is to find the target in optimal time [5]. Searching is central to many areas of computer science including data structures, computational geometry, and artificial intelligence. A version of the problem was first posed in 1963 by Bellman [12] and independently considered in 1964 by Beck [7], where the target was placed according to a known probability distribution on the real line, the robot was moving with uniform speed, and the goal was to find the target in minimum expected time.

In this paper, we consider the problem of *parallel, co-operative* search on the infinite line by n mobile robots at most f of which are faulty. The target is placed at a distance unknown to the robots. The robots start at the same time and location and can communicate instantaneously in wireless mode at any distance on the real line. While searching, the robots may co-operate by exchanging (broadcasting) messages; however, the search may be impeded by some of the robots (at most f) which may exhibit byzantine faults. The ultimate goal is to minimize the time it takes all non-faulty robots to be certain that the correct location of the target has been found.

1.1 Motion and communication model

To begin, we describe the robots' locomotive and communication models used in a search algorithm.

Robots and their trajectories. Robots are assumed to start at a common location, considered to be the origin of the line. They can move at maximum unit speed either along the positive direction (described as moving *right*) or along the negative direction (described as moving *left*); any robot can change direction arbitrarily often (by *turning*) without any loss in time. An algorithm for parallel search specifies a *trajectory* unique to each robot that is given by its turning points, and the speed(s) to follow between turning points. Since each robot has a distinct identity, it may also follow a distinct trajectory. Robots are assumed to have full knowledge of all trajectories, and moreover can communicate instantaneously with each other in wireless mode at any distance. Since robots know all the trajectories, the only kind of message broadcast by a robot R is whether or not it has found the target at some location; if R stays silent while visiting some location, the implicit assumption made by the other robots is that R did not detect the target there. Thus R follows its predefined trajectory until either it finds the target, in which case it announces that it has found the target, or it hears some other robot R' announce that it has found the target, at which point R may change its trajectory to participate in a verification protocol in regard to the announcement.

Messages and communication. All n robots know that f of the robots are faulty but they cannot differentiate in advance which among them are faulty; instead they must distinguish faulty from non-faulty ones based on conflict resolution and verification of messages received throughout the communication exchanges taking place during the execution of the search protocol. To this end, robots are equipped with pairwise distinct identities which they cannot alter at any time (in that respect our model is similar to the weakly Byzantine agent in [22]). In addition to the *correct* identity, the current location of a robot is automatically included in any broadcast message sent by the robot. Consequently, a faulty robot that does not follow its assigned trajectory can be immediately detected as faulty by the other robots, if it chooses to broadcast at some stage. In all other ways, faulty robots are indistinguishable from non-faulty (*reliable*) robots, except that the former can make *deliberate* positive and negative detection *errors* as follows. A non-faulty or *reliable* robot never lies when it has to confirm or deny the existence of the target at some location. Contrast this with *a faulty robot that may stay silent even when it detects or visits the target, or may claim that it has found the target when, in fact, it has not found it.* Thus, a reliable robot *cannot necessarily trust* an announcement that the target has been found, nor can it be certain that a location - visited silently by another robot - does not contain the target. In other words, the search for a target can terminate only after at least one robot that is *provably reliable* has visited the target and announced that it has been found. This requirement is critical to all our algorithms: if at some time, multiple robots make conflicting announcements at a location then the resulting (conflicting) *votes* can only be resolved if something is known about the number of reliable robots that participated in the vote. For instance, if three robots vote and it is known that two of them are reliable, then the majority vote would be the truth.

1.2 Preliminaries and notation

Consider a parallel search algorithm for a target located at distance d from the origin. First we define the search time of the algorithm and its corresponding competitive ratio.

▶ **Definition 1** (Search Time). Let $S_d(n, f)$ denote the time it takes for a search algorithm using a collection of n robots at most f of which are faulty, to find in parallel the location of a target placed at a distance d (unknown to the robots) from the starting position (the origin) of the robots on the line.

▶ **Definition 2** (Competitive Ratio). The corresponding *competitive ratio* is defined as $S_d(n, f)/d$, which is the ratio of the algorithm's search time and the lower bound d on the time taken by any algorithm for the problem.

For larger values of n and f, it will be more convenient to express our results in terms of the *density*, $\beta = \frac{f}{n}$, of faulty robots. This leads to the following definition.

▶ **Definition 3** (Asymptotic Competitive Search Ratio). Extend the definition of $S_d(n, f)$ above to non-integer values of n by replacing n with $\lceil n \rceil$ while the parameter f remains integral. Let $\beta = \frac{f}{n}$. Then

$$\hat{S}(\beta) = \min\ \{\alpha \mid \exists \text{ constant } c_\beta \text{ such that } \forall f > 0,\ S_d\left(f/\beta + c_\beta, f\right) \leq \alpha d\} \tag{1}$$

denotes the asymptotic competitive search ratio of any algorithm with search time $S_d(n, f)$.

Note that if $n \geq 4f + 2$, then in any partition of the robots into two groups each of size at least $2f + 1$, we will always have at least $f + 1$ reliable (non-faulty) robots per group. Therefore, an algorithm that sends the corresponding robots in the two groups in opposite directions is guaranteed to find the target in time d, because when the target is visited by one of the groups, a straightforward majority vote in the group confirms its presence reliably. Hence, $S_d(4f + 2, f) = d$, which is optimal. On the other hand, if $n \leq 2f$, there is no algorithm to complete the search: the f faulty robots may always completely disagree with the reliable ones, making it impossible to be certain of the location of the target. Therefore, in the sequel, we examine the interesting case where $2f + 1 \leq n \leq 4f + 1$.

Table 1 Upper and lower bounds on the search time $S_d(n, f)$ for a given number $n \leq 6$ of robots and faults $f = 1, 2$. Byz UB and Byz LB denote the known upper and lower bound for byzantine faults while Crash UB and Crash LB denote the known upper and lower bound for crash faults.

n, f	Byz. UB	Byz. LB	Crash-UB	Crash-LB
$3, 1$	$9d$	$3.93d$	$5.24d$	$3.76d$
$4, 1$	$3d$	$3d$	d	d
$5, 1$	$2d$	$2d$	d	d
$6, 1$	d	d	d	d
$5, 2$	$9d$	$3.57d$	$4.43d$	$3.57d$
$6, 2$	$4d$	$3d$	d	d

Table 2 Upper and lower bounds on the asymptotic competitive search ratio $\hat{S}(\beta)$ for various ranges of the density β. Note that for $\beta > \frac{1}{2}$ the search problem is impossible to solve.

β	$\leq \frac{1}{4}$	$(\frac{1}{4}, \frac{3}{10}]$	$(\frac{3}{10}, \frac{1}{3}]$	$(\frac{1}{3}, \frac{5}{14}]$	$(\frac{5}{14}, \frac{13}{34}]$	$(\frac{13}{34}, \frac{19}{46}]$	$(\frac{19}{46}, \frac{47}{110}]$	$(\frac{47}{110}, \frac{65}{146}]$	$(\frac{65}{146}, \frac{157}{396}]$	$(\frac{157}{396}, \frac{1}{2}]$
UB	1	2	3	3	4	5	6	7	8	9
LB	1	2	2	3	3	3	3	3	3	3

1.3 Our results

In Section 2, we are concerned mostly with upper bounds. Subsection 2.1 establishes the guiding principles for the design of algorithms.

We begin our study of upper bounds in Subsection 2.2 by establishing bounds for $S_d(n, f)$ for specific small values of n and f. These results are summarized in Table 1. For a comparison, we include in Table 1 known results on the search time for algorithms on the line with faulty robots that exhibit only *crash* faults [19], i.e., when the faulty robots never send any messages.

For larger values of n and f we express our results in terms of the density $\beta = \frac{f}{n}$ and show how to extend our algorithms from small values of n and f to this setting. Table 2 summarizes our results from Subsection 2.3.

Subsection 2.4 concludes Section 2 with several intriguing algorithms in that for densities $\frac{f}{n}$ between $\frac{3}{10}$ and $\frac{1}{3}$ the resulting search time is between $2d$ and $3d$. In Section 3, we derive two lower bounds on the search time. All missing proofs are found in the full version of the paper.

1.4 Related work

A search problem is usually seen as localization of a hidden target using searchers capable to move in the environment. It is an optimization question, usually attempting to minimize the time needed to complete the search. The question has been studied in numerous variations involving static or moving targets, one or many searchers, known or unknown environment, synchronous or asynchronous settings, different speed agents and many others (cf. [23]).

In several studies, when the environment is not known in advance, search implies exploration, often involving mapping and localizing searchers within the environment [2, 3, 21, 24, 26, 30]. However, even for the case of a known, simple environment like a line, there were several interesting studies attempting to optimize the search time. They started with the independent works of Bellman [12] and Beck [7], in which the authors attempted to minimize the competitive ratio in a stochastic setting. More exactly, they proved that time $9d$ is needed to guarantee finding the target situated at a (*a priori* unknown) distance d from the origin. Several other works on linear search followed (e.g. see [4, 7, 8, 9, 10, 11, 12]). More recently the search by a single searcher was studied for different models, e.g., when the turn cost was considered [20], when the bounds on the distance to the target are known in advance [14], and when the target was moving or for more general linear cost functions [13].

Most recently variants of linear search were studied for collections of collaborating searchers (robots). [16] considered linear group search, when the process is completed when the target is reached by the last robot visiting it. The robots collaborate attempting to minimize the group search time. However, [16] shows that having many robots does not help and the optimal search time is still bounded from below by $9d$. Group search using a pair of robots having distinct maximal speeds was studied in [6], in which techniques producing optimal search time were designed.

Fault tolerance was studied in distributed computing in various settings in the past (e.g., see [25, 28, 29]). However, the subject of unreliability was mainly for static components of the environment (e.g. network nodes or links), which was sometimes modelled by dynamically evolving environments (cf. [15, 27]). The malfunctions arising to mobile robots were investigated for various problems of gathering or pattern forming [1, 17, 22, 31] or patrolling [18]. Recently [19] investigated crash faults of robots performing linear search, where the time of finding the target by the first reliable robot was optimized. However, dealing with Byzantine agents is in general more tricky, requiring to identify and to refute the most malicious adversarial behavior (e.g., see [22]).

2 Upper Bounds

As already observed, if $n \geq 4f+2$, linear search can be performed optimally in time d, and no algorithm exists if $n \leq 2f$. Therefore, we consider below the case when $2f+1 \leq n \leq 4f+1$. Clearly, the robots can always stay together as a group, and perform the doubling zig-zag strategy that is optimal for a single robot and that has competitive ratio 9 [5, 7]. Since the reliable robots (at least $f+1$) are always in a majority, we are guaranteed to find the target. This yields the following upper bound:

▶ **Theorem 4.** *$S_d(n, f) \leq 9d$.*

In the remainder of this section, we provide upper bounds that, in general, are better than those suggested by Theorem 4 for the search problem. We do so by identifying and using some *guiding* principles to design search algorithms in the presence of faulty robots.

2.1 Principles for the design of algorithms

The general framework of our algorithms involves five basic principles, namely *Partition into Groups, Symmetry of Algorithms, Resolution of Conflicts, Simultaneous Announcements*, and *Computations by the Robots*, which we describe below in detail.

Partition into Groups. Depending on the ratio of faulty robots, we partition the robots into a certain number of groups. Two of the groups lead the exploration in opposite direction from the origin of the line. Further, each of these two groups will have at least $f+1$ robots so that at least some of the robots would announce the target when it is reached.

Symmetry of Algorithms. The algorithms are symmetric as far as left and right part of the line is concerned. We therefore typically discuss the behavior of the algorithm with respect to one side of the line only.

Resolution of Conflicts. If at any time there is an announcement of a target, the robots in the search groups stop until the claim is resolved. In the meantime, robots from some other group(s) move to resolve the claim. Once the claim is resolved, either the target is found and the robots stop, or a certain number of faulty robots is identified. From this time onward, the algorithm disregards any message from these faulty robots, effectively reducing the number of faulty robots to contend with, and the groups continue the search. Thus, each such announcement exposes more of the faulty robots, until eventually, we can be certain of a majority of robots in each search group being reliable, in which case the remaining search can be easily finished.

Simultaneous Announcements. When two announcements are being made at the same time, as usual with wireless transmissions, the algorithm deals only with one of them at a time, chosen arbitrarily. After the resolution of the first announcement is done and the search is possibly restarted, the robots redo their observation, and then the announcement is repeated if needed, thus taking into consideration the situation after the resolution of the first announcement. We show it does not influence the search time.

Computations by the Robots We assume that the time spent on calculations is negligible in comparison with the time spent in moving. Thus, we count only the time needed in movements of the robots until the target is found.

As indicated above, throughout the execution of the algorithms, conflicts will be resolved by voting. More precisely, we define $V(x,t)$ to be the *vote* of the robots about position x at time t. If y robots have claimed that the target is at x at or before time t, while z have claimed (by visiting and keeping silent) that it is not at x, then we say $V(x,t) = (y,z)$.

▶ **Definition 5** (Conflict)**.** We say there is a *conflict* at position x at time t if $V(x,t) = (y,z)$, with $0 < y, z \leq f$.

The following two simple observations are used extensively in the proofs in this section.

▶ **Lemma 6.** *Let $V(x,t) = (y,z)$, and let f be the number of faulty robots before time t. Then*

1. *If $y > f$ then the target is at position x and the search is concluded.*
2. *If $z > f$ then the target is not at position x and y new faulty robots have been identified at time t.*

▶ **Lemma 7.** *Suppose at time t, there are f' faulty robots remaining, and there are at least $2f'+1$ robots at positions $\geq x$ and at least $2f'+1$ robots at positions $\leq -x$. Then any target that is distance d from the origin can be found in time $t + (d-x)$.*

To build intuition, we start with giving algorithms with at most 2 faulty robots, and later show how to use these techniques to give algorithms with asymptotic ratios for general values of n and f.

2.2 Algorithms for $n \leq 6$

Since $2f + 1 \leq n < 4f + 2$, there are only two kinds of possible combinations of values with $n \leq 6$: either $f = 1$ and $3 \leq n \leq 5$, or $f = 2$ and $5 \leq n \leq 6$.

▶ **Proposition 8.** $S_d(4, 1) \leq 3d$

Proof of Proposition 8. Partition all robots into two search groups, L, and R, with two robots in each group. Each robot in R (L) moves right (left resp.) at speed 1 until it finds the target or hears an announcement that the target has been found. Suppose now that there is an announcement at time x from position $x > 0$. If $V(x, x) = (2, 0)$, by Lemma 6, the target has been found at x and the algorithm terminates. Suppose that $V(x, x) = (1, 1)$. Then one of the robots in L, say A, travels to x to resolve the conflict, taking additional time $2x$, while all other robots remain stationary. At time $3x$, the robot A reaches x. If $V(x, 3x) = (2, 1)$, by Lemma 6, the target has been found, and the algorithm terminates. If instead $V(x, 3x) = (1, 2)$, then by Lemma 6, the faulty robot is identified, and all other robots can be inferred to be reliable. Now the search continues with the groups moving in opposite directions with only the reliable robots being considered, until the target is found. Notice that an announcement at $-x$, simultaneous with that at x, would be resolved at time $3x$ with reliable robots. Therefore, if the target is at d or $-d$, the time taken to find it is $\leq 3x + d - x = 2x + d \leq 3d$ since $d > x$. Thus in all cases, $S_d(4, 1) \leq 3d$. ◀

If the number of robots increases to $n = 5$ (while f still equals 1), then it is possible to send two groups of size 2 in opposite directions as in the algorithm above, but keep one *spare* robot at the origin for conflict resolution. This improves the search time to at most $2d$ since the spare is always at a distance d from a conflicting vote, and moreover, the spare is definitely reliable since the faulty robot is part of the conflicting vote..

▶ **Proposition 9.** $S_d(5, 1) \leq 2d$

Note that the cases, $(n, f) = (5, 2)$ or $(n, f) = (3, 1)$, satisfy $n = 2f + 1$, the bare minimum of robots necessary to guarantee termination. For these cases, it seems very difficult to improve upon the upper bound on $S_d(n, f) \leq 9d$ from Theorem 4. In fact, we conjecture that this best possible for the pairs $(5, 2)$ and $(3, 1)$ stated above.

By ensuring an appropriate *redistribution* of robots past the announcement of a conflict, we can show the following result:

▶ **Proposition 10.** $S_d(6, 2) \leq 4d$.

2.3 Algorithms for large n

We now consider the case of large n, with different values of the density, $\beta = f/n$, of faulty robots. We start with generalizing the results from the previous subsection, then build recursive techniques that allow us to deal with larger densities of faulty robots, while paying a price in terms of the search time.

▶ **Theorem 11.** $S_d\left(\frac{10f+4}{3}, f\right) \leq 2d$, *provided that* $f \equiv 2 \mod 3$.

Using the fact that $S_d(n+k, f) \leq S_d(n, f)$ for any $k \geq 0$ and that $\hat{S}(\beta) \leq \hat{S}(\beta')$ if $\beta \leq \beta'$ we can easily derive the following corollary:

▶ **Corollary 12.** *If* $\beta \leq \frac{3}{10}$ *then* $\hat{S}(\beta) \leq 2$.

▶ **Theorem 13.** *$S_d\left(\frac{14f+4}{5}, f\right) \leq 3d$ provided $f \equiv 4 \mod 5$.*

Proof of Theorem 13. Partition the robots into two search groups L and R each containing $\frac{7f+2}{5}$ robots. The robots in L move left and those in R move right at speed 1. Without loss of generality, assume there is an announcement at x at time x. Let $V(x,x) = (y,z)$. Then if $\max\{y,z\} > f$, the announcement is resolved using Lemma 6. Suppose instead that $\max\{y,z\} \leq f$. Then $\min\{y,z\} \geq \frac{2f+2}{5}$ and at least $\frac{2f+2}{5}$ robots at x are faulty. In this case, $\frac{3f+3}{5}$ robots from L move from $-x$ to x, and at the same time $\frac{2f+2}{5}$ robots that voted yes and $\frac{2f+2}{5}$ that voted no are sent from x to $-x$. At time $3x$, in total $2f+1$ robots have voted at x, and by Lemma 6, either the target is identified, or at least $\frac{2f+2}{5}$ faulty robots are identified at $-x$ and may be disregarded from now on. There are at most $\frac{3f-2}{5}$ faulty robots unidentified. After the exchange of robots and elimination of the faulty robots in the worst case there are $\frac{6f+1}{5}$ robots in L and in R, i.e., a majority of reliable robots in both search groups. Therefore by Lemma 7, search for a target at distance d can be finished in time $3x + d - x \leq 3d$ as claimed. Note that all quantities are integral if $f \equiv 4 \mod 5$. ◀

As above, the following corollary is immediate:

▶ **Corollary 14.** *If $\beta \leq \frac{5}{14}$ then $\hat{S}(\beta) \leq 3$.*

As illustrated in the proofs of Theorems 11 and 13, when an announcement of a target is made, either the target can be confirmed, or the number of unidentified faulty robots can be reduced by an exchange of robots between the two search groups. For higher densities of faulty robots this technique can be repeated, for which we pay by an increase in the search time. This is the motivation for the recurrence formulas below that are used to obtain search algorithms for higher densities of robots.

▶ **Definition 15.** Let $T_x(l,s,r,f)$ be the minimum search time required by the robots to find the target given that initially, l robots are located at $-x$, s robots are at the origin 0, r robots are at $+x$, and f robots are faulty.

Since, as in the algorithms described so far, one way to solve our search problem is to send two equal-sized groups of robots to positions x and $-x$, we get the following upper bound.

▶ **Lemma 16.** *$\forall d \geq x > 0$ $S_d(n,f) \leq x + T_x(n/2, 0, n/2, f)$. Furthermore, if $n/2 \geq 2f+1$ then $T_x(n/2,0,n/2,f) = d - x$.*

If there is an announcement at x, we can identify some of the faulty robots, and by paying a price in terms of additional time, we can reduce it to a new problem with a smaller number of faulty robots. This can be encapsulated in the following lemma:

▶ **Lemma 17.** *Let $k > 0$ be even. Suppose there is an announcement at distance x from the origin. Then for all $a \geq x$, $T_x(f+k, 0, f+k, f) \leq 2x + T_a(f+k/2, 0, f+k/2, f-k)$.*

Proof of Lemma 17. Assume there are $f+k$ robots each at x and $-x$, with at most f faulty robots in all, and that a conflict occurs at $x > 0$ at some time t. Let $V(x,x) = (y,z)$. Then $k \leq \min\{y,z\} \leq \max\{y,z\} \leq f$. Now the robots move as follows:
1. All $f+k$ robots at position x move to $-x$.
2. $f+k/2$ of the robots at $-x$ move to x.

Note that these movements take time $2x$, and there are now $f+k/2$ robots at x and $f+k/2+k$ robots at $-x$. Since $2f+3k/2$ robots have now visited x, the vote $V(x,3x)$ is enough to resolve the conflict, and there remain at most $f-k$ faulty robots among the total $2f+k$ robots. This proves the lemma. ◀

Lemma 17 along with Theorem 13 can be used to obtain slower algorithms for higher densities. We have:

▶ **Theorem 18.**
1. $\hat{S}(\beta) \leq 5$ *for* $\beta \leq 19/46$.
2. $\hat{S}(\beta) \leq 7$ *for* $\beta \leq 65/146$.

Similar to Lemma 17 the following lemma establishes a recurrence that can be used to extend Theorem 11 to higher densities (at a cost of a higher competitive ratio).

▶ **Lemma 19.** *Suppose there is an announcement at distance x from the origin. Then for all $a \geq x$ and $k \geq f/4$: $T_x(f+k, 0, f+k, f) \leq 2x + T_a\left(\frac{4(f-k)}{3}, \frac{2(f-k)}{3}, 3k, f-k\right)$.*

Using a similar argument to that used in Theorem 18 we can apply Lemma 19 and Theorem 11 to get:

▶ **Theorem 20.**
1. $\hat{S}(\beta) \leq 4$ *for* $\beta \leq 13/34$.
2. $\hat{S}(\beta) \leq 6$ *for* $\beta \leq 47/110$.
3. $\hat{S}(\beta) \leq 8$ *for* $\beta \leq 157/396$.

2.4 Algorithms for $\frac{3}{10} \leq \beta < \frac{1}{3}$

Finally we discuss a new class of algorithms for densities of $\frac{f}{n}$ between $\frac{3}{10}$ and $\frac{1}{3}$ whose search time is between $2d$ and $3d$.

Informally, in any of these algorithms, the robots are partitioned into two search groups, that move in opposite directions at speed 1, and i middle groups, i odd, $i \geq 3$, positioned at regular intervals between the search groups. These i groups are used to solve any conflict reached by the search groups. The positioning of the middle groups between the search groups is achieved by them moving at a fraction of the maximal speed.

When a vote arises that cannot be resolved using Lemma 6, the middle groups are moved to the point of conflict in sequence at speed 1 until a resolution of the conflict is obtained. The middle groups not used in the resolution of a conflict on one side can be used to resolve a conflict on the other side. This approach allows a fine-grain resolution of a conflict by taking into account the result of the vote each time a group arrives to the conflict point.

▶ **Lemma 21.** *Let i be an odd integer, $i \geq 3$.*
$S_d(\frac{(3i+2)f}{i+1} + 2, f)) \leq \left(3 - \frac{2}{i+1}\right) d$, provided $f \equiv 0 \mod (i+1)$.

▶ **Corollary 22.**
1. $\hat{S}(\beta) \leq 2.5$ *for* $\beta \leq 4/13$.
2. $\hat{S}(\beta) \leq 2.67$ *for* $\beta \leq 6/19$.
3. $\hat{S}(\beta) \leq 2.75$ *for* $\beta \leq 8/25$.
4. $\hat{S}(\beta) \leq 2.8$ *for* $\beta \leq 10/31$.

3 Lower Bounds

It is straightforward to see that to achieve search time d, $4f + 2$ robots are necessary; with $4f + 1$ or fewer robots, at time d, either d or $-d$ can be visited by at most $2f$ robots. The adversary can make f of these $2f$ robots faulty, and it is impossible to be certain about the answer. Formally we can prove the following result.

▶ **Lemma 23.** $S_d(5,1) \geq 2d$.

Proof of Lemma 23. At time $d-\epsilon$ no one has visited d or $-d$. Consider where the robots are at this time. It must be the case that one of the intervals $(-d,0)$ or $(0,d)$ contains at most 2 robots. Without loss of generality say it is $(0,d)$. Put the target at d. Sort the robots by distance to d (ties broken arbitrarily) and make the robot closest to d faulty and silent. Then at least one robot from $(-d,0]$ must also reach d so that two non-faulty robots can identify the target at d. Thus, the search time is at least $d-\epsilon+d = 2d-\epsilon$. ◀

The next theorem shows that the density $f/n = \frac{3}{10}$ in Theorem 11 is also a lower bound on this ratio if we want to maintain the search time to be at most $2d$.

▶ **Theorem 24.** *If $S_d(n,f) \leq 2d$ then $\frac{f}{n} \leq \frac{3}{10}$.*

Proof of Theorem 24. Assume on the contrary that $\frac{n}{f} \leq \frac{10}{3}-\epsilon$ and that there is an algorithm for solving the search problem in time $2d$. Observe the intervals $[-d,0)$, $\{0\}$, $(0,+d]$ at time d and let us denote by l,r the number of robots within $[-d,0)$, $(0,+d]$, and by s the number of robots at the origin 0, respectively. By assumption $l+r+s = (10/3-\epsilon)f$. Observe that robots which are located at points different from $-d,0,d$ at time d may not be helpful in reducing the $2d$ search time. Thus, without loss of generality we may assume that at time d only the points $-d,0,+d$ are occupied by robots. Without loss of generality assume that $r \leq l$. We derive a contradiction by considering two cases.

1. Either l or $r \geq \frac{4}{3}f$. In this case we have that $r+s = \frac{10}{3}f-\epsilon-l \leq \frac{10}{3}f-\epsilon-\frac{4}{3}f = 2f-\epsilon$. Thus, $s+r$ robots are not sufficient to resolve conflicts on the right possibly involving f faulty robots within time $2d$.
2. Assume that there exists $\epsilon > 0$ such that both, $l,r \leq (\frac{4}{3}-\epsilon)f$. In particular, consider $r \leq (\frac{4}{3}-\epsilon)f$. Consider time d and suppose that up to $min\{r, \frac{1}{3}f\}$ of robots at d claim to find the target. For the algorithm to attain time $2d$, robots must be send from the start position 0 at time d to position d so as to verify the claim. Since among the robots sent to $+d$ from 0 we could have all remaining faulty robots, the number of robots sent from 0 must be at least $2f+1-r$ so that we a decision at time $2d$ can be made. However, if the target is not at $+d$ then the adversary could make it so that only $\frac{1}{3}f$ robots are faulty at $+d$ from among $2f+1$ robots. However, now we have at most $\frac{10}{3}f-\epsilon-2f-1 = \frac{4}{3}f-\epsilon-1$ robots at 0 or to the left of 0 and still $\frac{2}{3}f$ faulty robots remain among them. Thus, any claim of target at $-d'$ to the left of $-d$ cannot be verified in time $2d'$ by the available robots.

This proves the theorem. ◀

▶ **Lemma 25.** $S_d(3f+1,f) = 3d$.

Proof of Lemma 25. The upper bound $S_d(3f+1,f) \leq 3d$ has been proved in Theorem 13. To prove the lower bound $S_d(3f+1,f) \geq 3d$ we argue as follows. Consider visits to the set of symmetric positions $\{-d,+d\}$ by the robots. In particular, consider the first time t that at least $f+1$ robots complete visits to the *second* of the positions in the set. For instance, without loss of generality, assume that position $-d$ is visited first by at least $f+1$ robots and later (or instantaneously) by at least $f+1$ robots. Clearly the time t is at least d. The adversary arranges for a conflict at position $+d$. Note that unless $t \geq 3d$, the sets of robots visiting the two positions must be disjoint, and hence, the conflict at position $+d$ involves at most $2f$ robots participating in a vote, i.e. to resolve the conflict, at least one of the robots that visited $-d$ must move to $+d$. It follows that the total time required is at least $t+2d \geq 3d$. ◀

Note that Lemma 25 implies a lower bound for densities β in the range $1/3 > \beta > 3/10$. In case $n = 3,\ f = 1$, we can show the following lower bound on the search time.

▶ **Lemma 26.** *$S_d(3,1) \geq 3.93d$.*

Proof. (Lemma 26) We start by considering three positive real numbers x, y, α such that

$$\frac{\alpha-1}{2} \leq x < y \leq \frac{2}{\alpha-3} \text{ and } \frac{\alpha-1}{2} \leq \frac{y}{x} \leq \frac{2}{\alpha-3}. \tag{2}$$

We will show that an α satisfying Inequalities (2) above is the competitive ratio of all search algorithms for three robots with one Byzantine fault. Moreover, using Mathematica it can be shown that the maximum value of α that satisfies (2) is 3.93.

Consider numbers $-y,\ -x,\ -1,\ 0,\ 1,\ x,\ y$ on the real line and the movement of the three robots with respect to these points. Assume on the contrary the competitive ratio is some value ρ such that $\rho < \alpha$. Throughout the arguments below we are using Inequalities (2).

Observe that two robots must visit the points $-1, 1$ before time α, otherwise we get a contradiction to the competitive ratio because of Inequality (2). Therefore there exists a robot, say A, that visited both of these points before time α. Same argument applies for points $-x, x$. There exist a robot that visits both points $-x, x$ before time αx. Observe that this robot cannot be A. Indeed, otherwise it takes either time $2x + 1$ to reach point -1 or time $2 + 3x$ to reach point x. Let B be the robot that visits both points $-x, x$ before time αx. Because of the time constraints in Inequalities (2) the robot B must have either positive trajectory (i.e., visiting x before $-x$) or negative trajectory (i.e., visiting $-x$ before x). However, it is easy to see that B cannot have a positive trajectory because it would be too far to confirm an target placed at -1. This proves the lemma ◀

4 Discussion

In this paper, we considered a generalization of the well-known cow-path problem by having the search done in parallel with a group of n robots, with up to f of them being byzantine faulty. We presented optimal search algorithms for several ranges of values for $\beta = f/n$, the fraction of faulty robots, and gave non-trivial upper and lower bounds in many cases. Several interesting problems in the setting remain open, the most interesting one being to give tight upper and lower bounds in the case $n = 2f + 1$.

References

1 N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):56–82, 2006.
2 S. Albers and M. R. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164–1188, 2000.
3 S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32(1):123–143, 2002.
4 S. Alpern and S. Gal. *The theory of search games and rendezvous*, volume 55. Kluwer Academic Publishers, 2002.
5 R. Baeza Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.
6 E. Bampas, J. Czyzowicz, L. Gasieniec, D. Ilcinkas, R. Klasing, T. Kociumaka, and D. Pajak. Linear search by a pair of distinct-speed robots. In *SIROCCO*. to appear, 2016.
7 A. Beck. On the linear search problem. *Israel J. of Mathematics*, 2(4):221–228, 1964.

8 A. Beck. More on the linear search problem. *Israel J. of Mathematics*, 3(2):61–70, 1965.

9 A. Beck and M. Beck. Son of the linear search problem. *Israel Journal of Mathematics*, 48(2-3):109–122, 1984.

10 A. Beck and D. Newman. Yet more on the linear search problem. *Israel J. of Mathematics*, 8(4):419–429, 1970.

11 A. Beck and P. Warren. The return of the linear search problem. *Israel J. of Mathematics*, 14(2):169–183, 1973.

12 R. Bellman. An optimal search. *SIAM Review*, 5(3):274–274, 1963.

13 P. Bose and J.-L. De Carufel. A general framework for searching on a line. In *WALCOM: Algorithms and Computation – 10th International Workshop, WALCOM 2016, Kathmandu, Nepal, March 29-31, 2016, Proceedings*, pages 143–153, 2016.

14 P. Bose, J.-L. De Carufel, and S. Durocher. Revisiting the problem of searching on a line. In *Algorithms – ESA 2013 – 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 205–216, 2013.

15 A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. In *Ad-hoc, mobile, and wireless networks, LNCS*, volume 6811, pages 346–359. Springer, 2011.

16 M. Chrobak, L. Gasieniec, Gorry T., and R. Martin. Group search on the line. In *SOFSEM 2015*. Springer, 2015.

17 R. Cohen and D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM Journal of Computing*, 41(1):1516–1528, 2005.

18 J. Czyzowicz, L. Gasieniec, A. Kosowski, E. Kranakis, D. Krizanc, and N. Taleb. When patrolmen become corrupted: Monitoring a graph using faulty mobile robots. In *Algorithms and Computation – Proceedings of 26th ISAAC 2015*, pages 343–354, 2015.

19 J. Czyzowicz, E. Kranakis, D. Krizanc, L. Narayanan, and Opatrny J. Search on a line with faulty robots. In *PODC*. ACM, 2016.

20 E. D. Demaine, S. P. Fekete, and S. Gal. Online searching with turn cost. *Theoretical Computer Science*, 361(2):342–355, 2006.

21 X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment. In *FOCS*, pages 298–303. IEEE, 1991.

22 Y. Dieudonné, A. Pelc, and D. Peleg. Gathering despite mischief. *ACM Transactions on Algorithms (TALG)*, 11(1):1, 2014.

23 F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245, 2008.

24 F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM Journal on Computing*, 31(2):577–600, 2001.

25 J. Hromkovič, R. Klasing, B. Monien, and R. Peine. Dissemination of information in interconnection networks (broadcasting & gossiping). In *Combinatorial network theory*, pages 125–212. Springer, 1996.

26 J. Kleinberg. On-line search in a simple polygon. In *SODA*, page 8. SIAM, 1994.

27 F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 513–522. ACM, 2010.

28 L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

29 N. A. Lynch. *Distributed algorithms.* Morgan Kaufmann, 1996.

30 C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *ICALP*, pages 610–620. Springer, 1989.

31 S. Souissi, X. Défago, and M. Yamashita. Gathering asynchronous mobile robots with inaccurate compasses. *Principles of Distributed Systems*, pages 333–349, 2006.