Searching in the Presence of Linearly Bounded Errors

(Extended Abstract)

Javed A. Aslam*
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Aditi Dhagat†
Department of Mathematics
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

We consider the problem of determining an unknown quantity x by asking "yes-no" questions, where some of the answers may be erroneous. In particular, we focus on a *linearly bounded* model of errors where for some known constant r, $0 < r < \frac{1}{2}$, each initial sequence of i answers is guaranteed to have no more than ri errors. This model allows the errors to occur in a somewhat malicious way: for example, we must deal with a scenario where all errors occur in the last few answers. The problem is examined under the following variations: kinds of questions allowed (comparison or membership), nature of the domain of the searched quantity (bounded [$x \in \{1, \ldots, n\}$, for some fixed n] or unbounded [x is any positive integer]). In the bounded domain, the only previous bound on the number of questions which works for the entire range $0 < r < \frac{1}{2}$ (with either comparison or membership questions) is $O(n^{\log_2 \frac{1}{1-2r}})$. We improve this significantly by showing that $O(\log n)$ membership questions are enough. The upper bound on number of comparison questions needed is improved to $O(n^{\log_2 \frac{1}{1-r}})$, which is $o(n)$ even when r gets arbitrarily close to $\frac{1}{2}$. In the unbounded domain, where now n is the number being searched, we show that $O(\log n)$ membership questions or $O([n \log^2 n]^{\log_2 \frac{1}{1-r}})$ comparison questions are enough. The problem is solved using the framework of chip games.

*Author was supported by DARPA Contract N00014-87-K-825 and National Science Foundation Grant CCR-8914428. Author's net address: jaa@theory.lcs.mit.edu

†Author was supported by DARPA Contract N00014-87-K-825, National Science Foundation Grant CCR-8912586, and Air Force Contract AFOSR-89-0271. Author's net address: aditi@theory.lcs.mit.edu

1 Introduction

Coping with erroneous information during computation has been studied in various contexts: error-correctiong codes [Ber68], selection and sorting [FPRU90, RGL87], evaluating boolean functions [KY90], learning [AL86, KL88, GKS90], and searching [RMK$^+$80, Pel87, Pel88, Pel89a, Pel89b, RL84, Fra90], to name a few. In this paper we further examine this problem in the context of searching.

We consider the problem of determining an unknown quantity x by asking "yes-no" questions where some of the answers may be erroneous. This problem can be further qualified by:

- Kinds of questions that may be asked.
  - **Comparison questions**: "Is x less than y?"
  - **Membership questions**: "Is x in the set S?", where S is some subset of the domain.
- Kinds of errors possible.
  - **Constant number**: It is known *a priori* that there will be at most some k errors, where k is some fixed constant.
  - **Probabilistic**: The answer to each question is erroneous independently with some probability p, $0 < p < \frac{1}{2}$.
  - **Linearly Bounded**: For some constant r, $0 < r < \frac{1}{2}$, any initial sequence of i answers has at most ri errors. This model allows the answers to be erroneous in a malicious way. Unlikely scenarios in the probabilistic model (such as a long sequence of correct answers followed by a short sequence of false ones) must be dealt with here.
- Domain of searched quantity.
  - **Bounded**: $x \in \{1, \ldots, n\}$, for some known n.
  - **Unbounded**: x may be any positive integer.

	Membership Questions		Comparison Questions	
	$0 < r < \frac{1}{3}$	$\frac{1}{3} \le r < \frac{1}{2}$	$0 < r < \frac{1}{3}$	$\frac{1}{3} \le r < \frac{1}{2}$
Pelc	$O(\lg n)$	$O(n^{\lg \frac{1}{1-2r}})$	$O(\lg n)$	$O(n^{\lg \frac{1}{1-2r}})$
A-D	$O(\lg n)$		$O(n^{\lg \frac{1}{1-r}})$	

Figure 1: Bounds for searching in the bounded domain with linearly bounded errors. Here n is a bound on the number being sought.

	Membership Questions		Comparison Questions	
	$0 < r < \frac{1}{3}$	$\frac{1}{3} \le r < \frac{1}{2}$	$0 < r < \frac{1}{3}$	$\frac{1}{3} \le r < \frac{1}{2}$
Pelc	$O(\lg n)$	$O(n^{\lg \frac{1}{1-2r}})$	$O(\lg n)$	$O(n^{\lg \frac{1}{1-2r}})$
A-D	$O(\lg n)$		$O([n \lg^2 n]^{\lg \frac{1}{1-r}})$	

Figure 2: Bounds for searching in the unbounded domain with linearly bounded errors. Here n is the number being sought.

Work on this problem started with Rivest, et al. [RMK+80], who showed that in the bounded domain with comparison questions and a constant number of errors, x can be determined exactly in $O(\lg n)$ questions[1]. Naturally, this bound also applies to searching with membership questions, since comparison questions are a restricted version of membership questions.

In the probabilistic error model, Pelc [Pel89b] has shown that x can be determined correctly (within some given confidence of correctness) in $O(\lg n)$ questions if $p < \frac{1}{3}$, and in $O(\lg^2 n)$ questions if $\frac{1}{3} \le p < \frac{1}{2}$. These bounds apply to searching in both the bounded and unbounded domains.[2]

Pelc [Pel89b] has also examined the linearly bounded error model with comparison questions and shown that x can be determined exactly in $O(\lg n)$ questions in both the bounded and unbounded domains. However, these bounds only work for $r < \frac{1}{3}$. The best known bound with comparison questions in the bounded domain for $\frac{1}{3} \le r < \frac{1}{2}$ was $O(n^{\lg \frac{1}{1-2r}})$. Note that the degree of the polynomial in this bound becomes very large as r approaches $\frac{1}{2}$. This bound comes from a brute force binary search, where each question of the search is asked enough times so that the correct answer can be determined by taking majority. A simple argument [SW90, Fra90] shows that the search problem cannot be solved (with either membership or comparison questions) if $r \ge \frac{1}{2}$.

In this paper, we focus on the linearly bounded error model, where r may be any constant between 0 and $\frac{1}{2}$. We show that with membership questions, x can be determined exactly in $O(\lg n)$ questions in both the bounded and unbounded domains. These bounds are tight since searching has a trivial $\Omega(\lg n)$ lower bound. With comparison questions, we have improved the bounds to $O(n^{\lg \frac{1}{1-r}}) = o(n)$ questions for the bounded domain and $O([n \lg^2 n]^{\lg \frac{1}{1-r}}) = o(n)$ in the unbounded domain. A comparison of this work with best known results so far can be found in figures 1 and 2. Our results come from looking at the search problem in the framework of chip games. These chip games have also proved useful in modeling a hypergraph 2-coloring problem [AD90]. In general, chip games model computational problems in such a way that winning strategies for the players translate into bounds on the critical resource. This critical resource is represented by some aspect of the chip game, such as number of chips used or number of moves in the game.

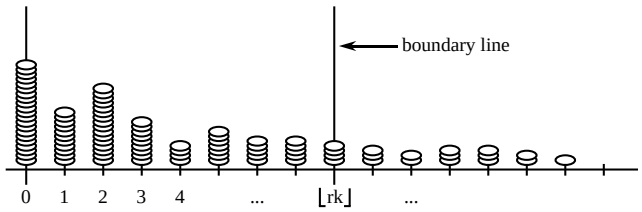

Figure 3: Chip Game

Spencer and Winkler [SW90] have also examined this problem. They have arrived independently at one of the theorems in this paper using different proof techniques.

We begin in section 2 by defining the searching problem with membership questions and the framework of chip games in which we solve it. In sections 3 and 4 we show an $O(\lg n)$ upper bound in both the bounded and unbounded domains. Section 5 examines the search problem with comparison questions and proves the aforementioned bounds. Section 6 concludes the paper.

2 Searching with Membership Questions and the Chip Game Formulation

Let $x \in \{1, \ldots, n\}$ be an unknown number. We wish to determine x by asking questions of the type "Is x in S?", for some $S \subseteq \{1, \ldots, n\}$. Some of the answers we receive may be erroneous. However, we are guaranteed that for some known constant r, $0 < r < \frac{1}{2}$, any initial sequence of i answers contains at most ri errors. How many questions are needed to determine x exactly?

This problem can be reformulated as a Chip Game between two players, the Pusher and the Chooser. The Chip Game starts with a unidimensional board marked in levels from 0 on upwards (see figure 3). We start with n chips on level 0, each chip representing one number in $\{1, \ldots, n\}$. At each step, the Chooser picks some chips

[1] $\lg n$ will denote $\log_2 n$ throughout this paper.

[2] In the unbounded domain, n refers to the number being sought.

from the board. These chips correspond to the subset S of $\{1, \ldots, n\}$ that we want to ask about. The Pusher now moves either the set of chips picked by the Chooser to the next level, indicating a "no" answer (i.e., x is *not* in S), or it does the same for the set of chips not picked by the Chooser, indicating a "yes" answer. Thus, the presence of a chip representing the number y at level i says that if y is the unknown number x, then there have been i lies about y. After some k steps, if a chip is at any level greater than $\lfloor rk \rfloor$, then it may be thrown away since the corresponding number can't possibly be the answer. To win, the Chooser must eliminate all but one chip from the board.

To clarify which chips may be thrown away, we will maintain a boundary line on the board. After k steps, the boundary line will be at level $\lfloor rk \rfloor$. Thus the Chooser may dispose of the chips at levels to the right of the boundary line at any time. Note that the boundary line moves one level to the right after approximately $\frac{1}{r}$ steps. The number of questions that we need to ask to determine x exactly is the same as the number of steps needed for the Chooser to win the above Chip Game.

3 $O(\lg n)$ questions are enough

We will show a winning strategy for the Chooser which requires $O(\lg n)$ steps. The strategy works in three stages. In stage 1, the Chooser eliminates all but $O(\lg n)$ chips from the board in $O(\lg n)$ steps. In stage 2, the Chooser eliminates all but $O(1)$ chips from the board in an additional $O(\lg n)$ steps. In stage 3, the Chooser removes all but one chip from the board in the final $O(\lg n)$ steps.

3.1 Stage 1

The strategy employed during stage 1 is simple. We describe it inductively on the number of steps as follows: let $h_m(i)$ be the height of the stack of chips at level i after m steps. In the $(m+1)$-st step, the Chooser will pick $\left\lfloor \frac{h_m(i)}{2} \right\rfloor$ from each stack of chips at all levels i. He will continue this way for $c_1 \lg n$ steps (where c_1 is a constant that will be determined in the analysis).

Before we can analyze this strategy, we will need a few definitions. Let *normalized binomial coefficients*

$$b_m(i) = \frac{n}{2^m} \binom{m}{i}$$

and let

$$\Delta_m(i) = h_m(i) - b_m(i).$$

The normalized binomial coefficient $b_m(i)$ will approximate $h_m(i)$, the height of the stack at level i after m steps, while $\Delta_m(i)$ will account for any discrepancy.

In order to analyze the given strategy, we need to be able to determine the number of chips which are to left of the boundary line after some number of steps in our strategy. After m steps, this is equivalent to $\sum_{i \le \lfloor rm \rfloor} h_m(i)$ (since r is the rate at which the boundary line moves). This sum is difficult to determine exactly. Instead, we will derive an upper bound for it by using the fact that $\sum_{i \le \lfloor rm \rfloor} h_m(i) = \sum_{i \le \lfloor rm \rfloor} b_m(i) + \sum_{i \le \lfloor rm \rfloor} \Delta_m(i)$. In particular, we will show an upper bound for $\sum_{i \le \lfloor rm \rfloor} \Delta_m(i)$.

For the strategy given above, we will now bound the discrepancy between the actual number of chips in any initial set of j stacks and the number of chips predicted by the normalized binomial coefficients. We will need three lemmas. The first two lemmas will handle boundary conditions, while the third will be required in the proof of the main theorem.

Lemma 1 $(\forall m \ge 0)$, $\Delta_m(0) \le 1$.

Proof: The proof is by induction on m.

- **base case:** For $m = 0$, $h_0(0) = n = b_0(0) \Longrightarrow \Delta_0(0) = 0$.
- **inductive step:** Assume $\Delta_{m-1}(0) \le 1$. We now have the following:

$$\begin{aligned} h_m(0) &\le \left\lceil \frac{h_{m-1}(0)}{2} \right\rceil \\ &\le \frac{h_{m-1}(0)}{2} + \frac{1}{2} \\ &= \frac{b_{m-1}(0)}{2} + \frac{\Delta_{m-1}(0)}{2} + \frac{1}{2} \\ &= b_m(0) + \frac{\Delta_{m-1}(0)}{2} + \frac{1}{2} \\ &\le b_m(0) + 1 \end{aligned}$$

Thus, $\Delta_m(0) = h_m(0) - b_m(0) \le 1$. ∎

Lemma 2 $(\forall m \ge 0)$, $\sum_{i=0}^{m} \Delta_m(i) = 0$.

Proof:

$$\sum_{i=0}^{m} h_m(i) = n = \sum_{i=0}^{m} b_m(i) \Longrightarrow \sum_{i=0}^{m} \Delta_m(i) = 0$$

∎

Lemma 3 $\sum_{i=0}^{j-1} b_{m-1}(i) + \frac{b_{m-1}(j)}{2} = \sum_{i=0}^{j} b_m(i)$.

Proof: We first note the fact that $\binom{a}{b-1} + \binom{a}{b} = \binom{a+1}{b}$. The proof proceeds as follows:

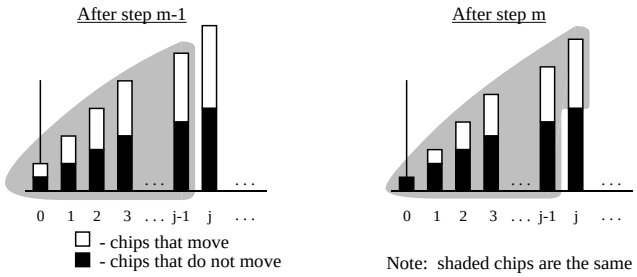

Figure 4: Chips before and after step m

$$\begin{aligned}
&\sum_{i=0}^{j-1} b_{m-1}(i) + \frac{b_{m-1}(j)}{2} \\
&= \sum_{i=0}^{j-1} \frac{n}{2^{m-1}}\binom{m-1}{i} + \frac{n}{2^m}\binom{m-1}{j} \\
&= \frac{n}{2^m}\left[\sum_{i=0}^{j-1} 2\cdot\binom{m-1}{i} + \binom{m-1}{j}\right] \\
&= \frac{n}{2^m}\left[\binom{m-1}{0} + \left[\binom{m-1}{0} + \binom{m-1}{1}\right] + \cdots \right. \\
&\quad \left. + \left[\binom{m-1}{j-1} + \binom{m-1}{j}\right]\right] \\
&= \frac{n}{2^m}\left[\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{j}\right] \\
&= \frac{n}{2^m}\sum_{i=0}^{j}\binom{m}{i} = \sum_{i=0}^{j} b_m(i)
\end{aligned}$$

■

Theorem 1 $(\forall m \geq 0)\ (\forall j \leq m),\ \sum_{i=0}^{j} \Delta_m(i) \leq j+1.$

Proof: The proof of the theorem is by induction on m. The base case of $m = 0$ is trivial. In the inductive step, we show that if the theorem holds for $m-1$, then the theorem holds for m. The boundary conditions $j = 0$ and $j = m$ are handled by Lemmas 1 and 2. We concentrate on the case $0 < j < m$ below. Consider the following (see figure 4):

$$\begin{aligned}
&\sum_{i=0}^{j} h_m(i) \\
&\leq \sum_{i=0}^{j-1} h_{m-1}(i) + \left\lceil \frac{h_{m-1}(j)}{2} \right\rceil \\
&\leq \sum_{i=0}^{j-1} h_{m-1}(i) + \frac{h_{m-1}(j)}{2} + \frac{1}{2} \\
&= \sum_{i=0}^{j-1} b_{m-1}(i) + \frac{b_{m-1}(j)}{2} + \\
&\quad \sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} \\
&= \sum_{i=0}^{j} b_m(i) + \sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2}
\end{aligned}$$

We will now bound the quantity $\sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2}$. There are two cases, depending upon whether $\Delta_{m-1}(j) \leq 1$ or $\Delta_{m-1}(j) > 1$. If $\Delta_{m-1}(j) \leq 1$, we have the following:

$$\begin{aligned}
\sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} &\leq \sum_{i=0}^{j-1} \Delta_{m-1}(i) + 1 \\
&\leq j+1
\end{aligned}$$

If $\Delta_{m-1}(j) > 1$, then $\frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} < \Delta_{m-1}(j)$. We thus obtain the following:

$$\begin{aligned}
\sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} &< \sum_{i=0}^{j} \Delta_{m-1}(i) \\
&\leq j+1
\end{aligned}$$

We therefore have

$$\begin{aligned}
&\sum_{i=0}^{j} h_m(i) \\
&\leq \sum_{i=0}^{j} b_m(i) + \sum_{i=0}^{j-1} \Delta_{m-1}(i) + \frac{\Delta_{m-1}(j)}{2} + \frac{1}{2} \\
&\leq \sum_{i=0}^{j} b_m(i) + j + 1
\end{aligned}$$

which implies that $\sum_{i=0}^{j} \Delta_m(i) \leq j+1$. ■

Now we will bound $\sum_{i \leq \lfloor rm \rfloor} b_m(i)$. We will find a constant c_1 such that for $m = c_1 \lg n$, $\sum_{i=0}^{\lfloor rm \rfloor} b_m(i)$ is a constant. If we can do this, then it follows from the theorem above that $\sum_{i=0}^{\lfloor rm \rfloor} h_m(i)$, the number of chips remaining to the left of the boundary line, is $O(\lg n)$. The reasoning goes thus:

$$\begin{aligned}
\sum_{i=0}^{\lfloor rm \rfloor} h_m(i) &= \sum_{i=0}^{\lfloor rm \rfloor} b_m(i) + \sum_{i=0}^{\lfloor rm \rfloor} \Delta_m(i) \\
&\leq \sum_{i=0}^{\lfloor rm \rfloor} b_m(i) + \lfloor rm \rfloor + 1 \\
&= c_2 + \lfloor r \cdot c_1 \lg n \rfloor \\
&\leq c_3 \lg n
\end{aligned}$$

for appropriate constants c_2 and c_3.

To determine c_1, note that:

$$\begin{aligned}
\sum_{i=0}^{\lfloor rm \rfloor} b_m(i) &= \frac{n}{2^m} \sum_{i=0}^{\lfloor rm \rfloor} \binom{m}{i} \\
&\leq \frac{n}{2^m} 2^{mH(r)} \\
&= n 2^{m(H(r)-1)}
\end{aligned}$$

where $H(r)$ is the binary entropy function[3].

This last quantity is $O(1)$ when $m = \frac{\lg n}{1-H(r)}$. Thus if we pick $c_1 = \frac{1}{1-H(r)}$, then after $m = c_1 \lg n$ steps, there will be at most $c_3 \lg n$ chips remaining on the board to the left of the boundary line.

The strategy in this stage can also be applied to the game where the boundary line starts out at level $l = O(\lg n)$ instead of at $l = 0$. We omit the proof in this extended abstract, but it can be shown that stage 1 still ends in $O(\lg n)$ steps with at most $O(\lg n)$ chips to the left of the boundary line. This fact will be useful when we examine the unbounded domain.

3.2 Stage 2

At the end of stage 1 we are left with some $c_2 \lg n$ chips on the board with the boundary line at level $c_1 \lg n$ (for appropriate constants c_1 and c_2). After stage 2, the Chooser will leave the board with $O(1)$ chips to the left of the boundary line after $O(\lg n)$ additional steps.

Before starting stage 2, we alter the board by moving everything on the board (chips *and* boundary line) to the right by $c_2 \lg n$, so that the boundary line is now at level $(c_1 + c_2) \lg n = c \lg n$. While this new board corresponds to a different game than the one we have played until now (i.e. to a game where many more questions and lies have occured), they are equivalent in the sense that the Chooser can win from the first board within k extra moves if and only if he can win from the second board within k extra moves.

Now move the chips to the left in such a way such that there is exactly one chip on each of the first $c_2 \lg n$ levels. Note that the Chooser does not help himself by doing this, since moving chips to the left is in effect ignoring lies that he knows about.

At each step in this stage, the Chooser will first order the chips from left to right, ordering chips on the same level arbitrarily. Then he will pick every other chip according to this order; that is, he will pick the 1st, 3rd, 5th, ... chips. We shall say that the board is in a *nice* state if no level has more than 2 chips.

Lemma 4 *Throughout stage 2, the board is in a* nice *state.*

Proof: We show this by induction on the number of steps in stage 2. Certainly at the beginning of stage 2, the board is in a *nice* state since each level is occupied by at most one chip. Now suppose that the board is in a *nice* state after i steps. Consider any level j after the $(i+1)$-st step. Since both levels $j-1$ and j had at most 2 chips before the $(i+1)$-st step, after this step level j retains at most one chip and gains at most one chip, thus ending with at most 2 chips. ∎

[3] $H(r) = -r \lg r - (1-r) \lg(1-r)$

Now we will show that after $O(\lg n)$ steps, there will be at most $2k$ chips remaining to the left of the boundary line. Here k is a constant (depending only on r) which we will determine later. If there are fewer than $2k$ chips to the left of boundary line, stage 2 terminates. Let the weight of a chip be the level it is on, and let the weight of the board be the weight of its $2k$ leftmost chips.

Lemma 5 *After each step in stage 2, the weight of the board increases by at least* $k-1$.

Proof: Of the $2k$ leftmost chips after step i, at least the leftmost $(2k-1)$ chips remain in the set of leftmost $2k$ chips after step $i+1$. (The $2k$-th chip may be on the same level as the $(2k+1)$-st. In this case, if the $2k$-th chip moves in step $i+1$, then the $(2k+1)$-st chip becomes the new $2k$-th chip.) At least $\lfloor \frac{2k-1}{2} \rfloor = k-1$ of these chips move to the right one level during step $i+1$, thus increasing the weight of the board by at least $k-1$. ∎

Let S be the number of steps taken during this stage and W be the weight of the board at the end of these S steps. Since the weight of the board goes up by at least $k-1$ at each step, and since the initial weight of the board was non-negative, $W \geq (k-1)S$. At the end of the S steps, the boundary line is at $c \lg n + \lfloor rS \rfloor$. So $W \leq 2k(c \lg n + rS)$. Putting these two inequalities together, we obtain:

$$\begin{array}{rrcl} & 2k(c \lg n + rS) & \geq & S(k-1) \\ \Rightarrow & 2kc \lg n & \geq & S(k-1-2kr) \\ \Rightarrow & \frac{2kc}{k-1-2kr} \lg n & \geq & S \end{array}$$

If we let $k = \frac{2}{1-2r}$, then $S \leq \frac{4c}{1-2r} \lg n = O(\lg n)$. Thus after $O(\lg n)$ steps, stage 2 ends leaving at most $2k$ chips to the left of the boundary line.

3.3 Stage 3

At the beginning of stage 3, the Chooser will move all of the remaining chips to level 0. Again, this is legal, since he is essentially choosing to ignore some information he has gathered.

The Chooser plays this stage in phases. Each phase corresponds to getting the correct answer to a single question. This is done by asking the question repeatedly until the majority of the answers given to this question must be true. In the chip game, at the beginning of each phase there will be a single stack of chips somewhere on the board. The Chooser will pick half of these chips. The selected half-stack corresponds to the question whose correct answer he wishes to determine. He continues picking the same half-stack throughout this phase until either it or the other half-stack moves beyond the boundary line. Then he begins the next phase with the remaining half stack. This will continue until there is only one chip left on the board to the left of the boundary line. Note that if there are m chips on the

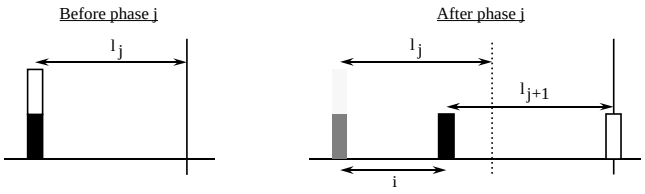

Figure 5: Chips before and after phase j.

board initially, $\lceil \lg m \rceil$ questions need to be answered correctly.

In what follows, we will show that if we start with d questions we wish to have answered correctly and the boundary line at a distance $l_0 > 0$ away from the single stack of chips on the board, then the Chooser can effect a win (i.e. remove all but one chip from the board) within $O(l_0(\frac{1}{1-r})^d)$ steps. Since stage 3 begins with some $2k$ chips a distance $c \lg n$ away from the boundary line (for appropriate constants c and k), the Chooser can win the game in $O[(c \lg n) \cdot (\frac{1}{1-r})^{\lceil \lg 2k \rceil}] = O(\lg n)$ steps in stage 3. Since each of the three stages takes $O(\lg n)$ steps, we will have shown that:

Theorem 2 *The problem of searching in the linearly bounded error model in the bounded domain $\{1, \ldots, n\}$ with membership questions and error constant r, $0 < r < \frac{1}{2}$, can be solved with $O(\lg n)$ questions.*

Now consider the board before and after some phase j. At the beginning of phase j, there is a stack of size m at some level some distance l_j away from the boundary line (see figure 5). At the end of phase j, one half-stack has moved to some distance i from its original position and the other has moved to the boundary line. The boundary line is now at some distance l_{j+1} from the first half-stack.

Let $T(d, l)$ be the number of steps the Chooser takes to answer d questions correctly when a single stack of chips on the board is a distance l away from the boundary line. We will make a couple of assumptions which simplify the analysis, without affecting our bounds for $T(d, l)$. First, we will assume that a stack may be discarded as soon as it reaches the line. Second, we will assume that the number of steps during phase j is exactly equal to the number of levels the boundary line moves divided by r. The number of actual steps may be at most $\lfloor \frac{1}{r} \rfloor$ more than this quantity. Now:

$$T(d, l_j) \leq T(d-1, l_{j+1}) + (\text{steps during phase } j)$$

Lemma 6 *Total number of steps during phase j is $\frac{2l_j - l_{j+1}}{1-2r}$.*

Proof: The total number of steps during phase j is the same as the total number of levels the two half-stacks move. One half-stack moves i levels and the other moves $i + l_{j+1}$ levels. The number of steps is also (approximately) equal to $\frac{1}{r}$(# levels the boundary line moves). Equating these two quantities, we obtain

$$i + i + l_{j+1} = \frac{i + l_{j+1} - l_j}{r}.$$

Solving the above equation for i, we get $i = \frac{l_j - (1-r)l_{j+1}}{1-2r}$. Substituting this value for i in the left hand side of the equation above, we have that the number of steps in phase j is $\frac{2l_j - l_{j+1}}{1-2r}$. ∎

Now we are ready to show that:

Theorem 3 $T(d, l_0) = \begin{cases} O(l_0(\frac{1}{1-r})^d) & \text{if } l_0 > 0 \\ O((\frac{1}{1-r})^d) & \text{if } l_0 = 0 \end{cases}$

Proof: We will only show the case for $l_0 > 0$. There are some technicalities involved in showing the $l_0 = 0$ case which we will not include in this extended abstract.

We first show that $\forall j, l_{j+1} \leq \frac{l_j}{1-r}$. Solving the equation:

$$i + i + l_{j+1} = \frac{i + l_{j+1} - l_j}{r}.$$

for l_{j+1}, we get:

$$\begin{aligned} l_{j+1} &= \frac{l_j + i(2r-1)}{1-r} \\ &\leq \frac{l_j}{1-r}, \text{ since } 2r-1 < 0 \text{ and } i \geq 0 \end{aligned}$$

We use this to find bounds on $T(d, l_0)$:

$$\begin{aligned} T(d, l_0) &\leq T(d-1, l_1) + \frac{2l_0 - l_1}{1-2r} \\ &= \frac{2l_0 - l_1}{1-2r} + \frac{2l_1 - l_2}{1-2r} + \cdots + \frac{2l_{d-1} - l_d}{1-2r} \\ &\leq \frac{1}{1-2r}[2l_0 + l_1 + l_2 + \cdots + l_{d-1}] \\ &\leq \frac{1}{1-2r}\left[2l_0 + \left(\frac{1}{1-r}\right)l_0 + \left(\frac{1}{1-r}\right)^2 l_0 \right. \\ &\quad \left. + \cdots + \left(\frac{1}{1-r}\right)^{d-1} l_0\right] \\ &= \frac{1}{1-2r}\left[l_0 + \frac{(\frac{1}{1-r})^d - 1}{\frac{1}{1-r} - 1} l_0\right] \\ &= O(l_0(\tfrac{1}{1-r})^d) \end{aligned}$$

∎

4 Unbounded Search

Now consider the problem of searching for a positive integer in the presence of errors as before, but where no upper bound on its size is known. Let this integer be n. Using strategies developed in this paper already, we will show that n can be found with $O(\lg n)$ questions.

The search will occur in two stages. First, we will determine a bound for the unknown number n. Second, given a bound on n, we will employ the techniques for bounded searching given above.

4.1 Stage 1

Consider the problem of bounding the unknown number n if all of the answers we receive are known to be correct. We could ask questions of the form "Is $x < 2^{2^i}$?". We would begin by asking "Is $x < 2^{2^0}$?". If the answer were "no", we would follow with "Is $x < 2^{2^1}$?", and so on. Since $n \leq 2^{2^{\lceil \lg \lg n \rceil}}$, we will obtain our first "yes" answer (and thus have a bound on n) after at most $\lceil \lg \lg n \rceil$ questions. We further note that our bound is not too large:

$$2^{2^{\lceil \lg \lg n \rceil}} < 2^{2^{\lg \lg n + 1}} = 2^{2 \lg n} = n^2$$

Employing the techniques and results of section 3.3, we can use the above strategy in the presence of errors. We need the correct answers to $\lceil \lg \lg n \rceil$ questions. By Theorem 3, we can obtain these answers in

$$O((\tfrac{1}{1-r})^{\lceil \lg \lg n \rceil}) = O((\lg n)^{\lg \frac{1}{1-r}}) = o(\lg n)$$

questions.

4.2 Stage 2

Having found a bound for n, we have reduced our unbounded search problem to a bounded search problem. We can now apply our bounded search strategy of section 3. It is important to note that since we have already asked $o(\lg n)$ questions, the boundary line will have moved to $o(\lg n)$. But recall that stage 1 of our bounded search algorithm can tolerate starting out with the boundary line at $O(\lg n)$. Thus, in this stage, the Chooser can start with all relevant chips at level 0 and boundary line at level $o(\lg n)$ and apply the bounded search strategy of section 3. Since our bound on the unknown number n is at most n^2, we will finish this stage after $O(\lg(n^2)) = O(\lg n)$ questions. We can now claim:

Theorem 4 *The problem of searching in the linearly bounded error model in the unbounded domain with membership questions and error constant r, $0 < r < \frac{1}{2}$ can be solved with $O(\lg n)$ questions, where the number being sought is n.*

5 Searching with Comparison Questions

In the bounded search problem, if the questions that the first player can ask are restricted to be comparison questions ("Is x less than y?"), then the Chooser can no longer pick arbitrary sets of chips as he does in stages 1 and 2 of the bounded case given in section 3.

However, the strategy in stage 3 of the bounded case lends itself well to comparison questions. Starting with n chips on the board at level 0, we will simply ask questions corresponding to a simple binary search. This search will require the correct answers to $\lceil \lg n \rceil$ questions. By Theorem 3, we can obtain these answers in

$$O((\tfrac{1}{1-r})^{\lceil \lg n \rceil}) = O(n^{\lg \frac{1}{1-r}}) = o(n)$$

questions.

Theorem 5 *The problem of searching in the linearly bounded error model in the bounded domain $\{1, \ldots, n\}$ with comparison questions and error constant r, $0 \leq r < \frac{1}{2}$, can be solved with $O(n^{\lg \frac{1}{1-r}}) = o(n)$ questions.*

We can employ techniques similar to those used above to solve the unbounded search problem using comparison questions. The first stage, in which a bound for the unknown number n is sought, will be identical to stage 1 of the unbounded search algorithm for membership questions given in section 4.1. We will thus bound the unknown number n by at most n^2 using $O((\lg n)^{\lg \frac{1}{1-r}})$ questions. Note that the boundary line will now be at $O((\lg n)^{\lg \frac{1}{1-r}})$.

Having bounded the unknown number n by at most n^2, we could simply use Theorem 3 directly. By performing a simple binary search, we will need the correct answers to at most $\lceil \lg(n^2) \rceil$ questions. Using Theorem 3, we obtain an overall question bound of

$$O((\lg n)^{\lg \frac{1}{1-r}} \cdot (\tfrac{1}{1-r})^{\lceil \lg(n^2) \rceil}) = O([n^2 \lg n]^{\lg \frac{1}{1-r}}).$$

This can be improved, however, by adding an extra stage. After bounding the unknown number n by at most n^2, partition this bounded interval into exponentially growing subintervals $I_j = [2^j, 2^{j+1} - 1]$ $\forall j \geq 0$. Note that there will be at most $\lceil \lg(n^2) \rceil$ such subintervals. To determine the correct subinterval, we will perform a simple binary search on these subintervals requiring the correct answers to $\lceil \lg \lceil \lg(n^2) \rceil \rceil$ questions. By Theorem 3, we will need

$$O((\lg n)^{\lg \frac{1}{1-r}} \cdot (\tfrac{1}{1-r})^{\lceil \lg \lceil \lg(n^2) \rceil \rceil}) = O([\lg^2 n]^{\lg \frac{1}{1-r}})$$

additional questions. Since our subintervals grew exponentially, the subinterval containing the unknown number n will be of size at most n. We can thus perform a final binary search on this subinterval, and employ Theorem 3 to obtain an overall question bound of

$$O([\lg^2 n]^{\lg \frac{1}{1-r}} \cdot (\tfrac{1}{1-r})^{\lceil \lg n \rceil}) = O([n \lg^2 n]^{\lg \frac{1}{1-r}}) = o(n).$$

Theorem 6 *The problem of searching in the linearly bounded error model in the unbounded domain with comparison questions and error constant r, $0 < r < \frac{1}{2}$ can be solved with $O([n \lg^2 n]^{\lg \frac{1}{1-r}}) = o(n)$ questions, where the number being sought is n.*

6 Conclusion

We have looked at the problem of searching in the presence of linearly bounded errors. We have shown that if membership questions are allowed, then $O(\lg n)$ questions are enough to determine x exactly in both bounded and unbounded domain. When comparison questions are allowed, we show that $O(n^{\lg \frac{1}{1-r}})$ and $O([n \lg^2 n]^{\lg \frac{1}{1-r}})$ questions are enough in the bounded and unbounded domains, respectively.

The upper bounds with membership questions are tight, since searching has a trivial $\Omega(\lg n)$ lower bound. The bounds with comparison questions can probably be improved.

References

[AD90] Javed A. Aslam and Aditi Dhagat. Online algorithms for 2-coloring hypergraphs via chip games. Technical Report MIT/LCS/TM-439, MIT Laboratory for Computer Science, 1990.

[AL86] D. Angluin and P. Laird. Identifying k-cnf formulas from noisy examples. Technical Report YALEU/DCS/TR-478, Yale University, 1986.

[Ber68] E. Berlekamp. *Error Correcting Codes*, pages 61–85. Wiley, N.Y., 1968.

[FPRU90] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Computing with unreliable information. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 128–137, 1990.

[Fra90] Michael Frazier. Searching with a non-constant number of lies. Unpublished manuscript, 1990.

[GKS90] S. Goldman, M. Kearns, and R. Schapire. Exact identification of circuits using fixed points of amplification functions. In *31st Symposium on Foundations of Computer Science*, 1990.

[KL88] Michael Kearns and Ming Li. Learning in the presence of malicious errors. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 267–280, 1988.

[KY90] Claire Kenyon and Andrew C. Yao. On evaluation of boolean functions with unreliable tests. *International Journal of Foundations of Computer Science*, 1(1):1–10, 1990.

[Pel87] Andrzej Pelc. Solution of Ulam's problem on searching with a lie. *Journal of Combinatorial Theory, Ser. A*, 44:129–140, 1987.

[Pel88] Andrzej Pelc. Prefix search with a lie. *Journal of Combinatorial Theory, Ser. A*, 48:165–173, 1988.

[Pel89a] Andrzej Pelc. Detecting errors in searching games. *Journal of Combinatorial Theory, Ser. A*, 41:43–54, 1989.

[Pel89b] Andrzej Pelc. Searching wih known error probability. *Theoretical Computer Science*, 63:185–202, 1989.

[RGL87] B. Ravikumar, K. Ganesan, and K. B. Lakshmanan. On selecting the largest element in spite of erroneous information. *Lecture Notes in Computer Science - ICALP*, pages 88–99, 1987.

[RL84] B. Ravikumar and K. B. Lakshmanan. Coping with known patterns of lies in a search game. *Theoretical Computer Science*, 33:85–94, 1984.

[RMK+80] R. L. Rivest, A. R. Meyer, D. J. Kleitman, K. Winklmann, and J. Spencer. Coping with errors in binary search procedures. *Journal of Computer and System Sciences*, 20:396–404, 1980.

[SW90] Joel Spencer and Peter Winkler. Three thresholds for a liar. Preprint, 1990.