Three Thresholds for a Liar

Joel Spencer*
NYU-Courant Institute
251 Mercer St.
New York, NY 10012

Peter Winkler†
Bellcore
445 South ST.
Morristown, NJ 07960-1910

November 5, 1991

A mixture of a lie doth ever add pleasure. -Francis Bacon 1561-1626

Abstract

Motivated by the problem of making correct computations from partly false information, we study a corruption of the classic game "Twenty Questions" in which the player who answers the yes-or-no questions is permitted to *lie* up to a fixed fraction r of the time. The other player is allowed q arbitrary questions with which to try to determine, with certainty, which of n objects his opponent has in mind; he "wins" if he can always do so, and "wins quickly" if he can do so using only $O(\log n)$ questions.

It turns out that there is a threshold value for r below which the querier can win quickly, and above which he cannot win at all. However, the threshold value varies according to the precise rules of the game. Our "three thresholds theorem" says that when the answerer is forbidden at any point to have answered more than a fraction r of the questions incorrectly, then the threshold value is $r = \frac{1}{2}$; when the requirement is merely that the total number of lies cannot exceed rq, the threshold is $\frac{1}{3}$; and finally if the answerer gets to see all the questions before answering, the threshold drops to $\frac{1}{4}$.

*Author's net address: spencer@cs.nyu.edu
†Author's net address: pw@bellcore.com

The Game

The problem of computing from unreliable information has attracted much interest in recent years. A typical model (see, for example, [FPRU90]) involves an oracle which, each time it is queried, responds falsely independently with probability some fixed p. Then the object is to design an algorithm which performs the required task correctly with probability arbitrarily close to 1, and with complexity not greatly exceeding that needed for a reliable oracle.

There are several potential difficulties with this model; among them are that independence can often not be assumed, and that even when conditions are ideal the output will with non-zero probability be wrong. We can eliminate those difficulties (replacing them, of course, with others) by trying instead to design algorithms which are *guaranteed* to produce correct output, provided no more than some fixed fraction r of the oracle's answers are false.

Worst-case analysis in this model puts the oracle in the role of a clever adversary, even less realistic an assumption, in most situations, than independence. However, if efficient algorithms which foil such an oracle *can* be found then of course less diabolical oracles with any probabilistic behavior are also taken into account.

Among the most fundamental of computations is the determination of a value from among n possible values, using binary queries, as in the classic game "Twenty Questions." It is this game which we will study, under altered rules which permit the player who answers the questions to lie—but not more than some fixed fraction r of the time.

We will dub the players "Paul" for the asker and "Carole" for the answerer, preserving the initials of the designations "Pusher" and "Chooser" from [Spe86]. The reader's memory may be aided by noting that "Carole" is an anagram of "oracle", and by thinking of Paul as the great questioner, Paul Erdős.

In any case, the game proceeds as follows: Carole thinks of a value v between 1 and n; Paul tries to determine v using questions of the form "is $v \in S$?" where S is an arbitrary subset of {1,...,n}.

For fixed r, n and q, a *winning strategy* for Paul is a scheme by which he can determine Carole's value, with certainty, using at most q questions (Carole's precise constraints are discussed below). Fixing only r, we say that Paul *wins* if for every n there is a q such that Paul has a winning strategy for the r, n, q game. We say that Paul wins *quickly* if he can win with $O(\log n)$ questions, that is, if there is a constant c_r such that Paul can win with $q = c_r \lg n$. Of course if $r = 0$ then we have the classic game which Paul wins (quickly) with precisely $\lceil \log_2 n \rceil$ questions.

Since Paul is required to succeed regardless of Carole's value, we may as well assume Carole does not need to commit to a value; a winning strategy for her is one which guarantees that after Paul's questions have been answered, he will always find himself with at least two values consistent with the constraints on Carole's answers. We thus have a perfect-information game. If for fixed r there is an n such that Paul has no winning strategy for any (finite) q, then we say that Carole wins.

At this point the reader may already have noticed that there is more than one possible set of ground rules for our game. In fact we consider three versions, successively less advantageous to Paul.

In Version A, Paul presents his questions one at a time and Carole must answer them in such a way that at no point has she lied to more than a fraction r of them; that is, every initial segment of a of her answers must contain no more than ra lies. Thus, for example, if $r < \frac{1}{3}$ then her first three answers must all be truthful.

In Version B, Paul again presents his questions one at a time but Carole is required only to make sure that at most rq of her q answers are lies. Here she may exhaust all her lies at the beginning if she wishes.

In Version C, Paul must submit all his questions simultaneously to Carole, and she is permitted to look them over before choosing a set of at most rq of them to lie to.

It is not immediately clear (at least, it wasn't to us) that the choice of rules actually makes any difference, but our results, collected in the following "Three Thresholds Theorem", show that this is indeed the case.

Theorem.

In Version A, Paul wins with $\Theta(\log n)$ questions if $r < \frac{1}{2}$ but Carole wins if $r \geq \frac{1}{2}$.
In Version B, Paul wins with $\Theta(\log n)$ questions if $r < \frac{1}{3}$ but Carole wins if $r \geq \frac{1}{3}$.
In Version C, Paul wins with $\Theta(\log n)$ questions if $r < \frac{1}{4}$; Carole wins if $r > \frac{1}{4}$; and Paul wins when $r = \frac{1}{4}$ exactly, but $\Theta(n)$ questions are required.

Let us make some remarks before proceeding to the proof. First, the constants implied in the Θ's do depend on r and in all three cases approach infinity as r approaches the threshold from below. At or above the threshold in Versions A and B, $n = 3$ and respectively $n = 5$ are already too many possibilities for Paul to handle with any number of questions; but in Version C the critical value of n approaches infinity as r approaches $\frac{1}{4}$ from above.

The problem of how to play Twenty Questions with a liar has been considered before, and various parts of the above theorem have been proved by others. In his autobiography "The Adventures of a Mathematician" [Ula76], Stanislaw Ulam writes:

> [David] Hawkins and I have speculated on the following problem: a variation on the game of Twenty Questions. Someone thinks of a number which is between one and one million (which is just less than 2^{20}). Another person is allowed to ask up to twenty questions, to each of which the first person is supposed to answer only yes or no. Obviously the number can be guessed by asking first: Is the number in the first half million? then again reduce the reservoir of numbers in the next question by one-half, and so on. Finally the number is obtained in less than log base 2 of 1,000,000. Now suppose one were allowed to lie once or twice, then how many questions would one need to get the right answer? One clearly needs more than n questions for guessing one of 2^n objects because one does not know when the lie was told. This problem is not solved in general.

Various versions of the game, some probabilistic or involving queries of special form, have been considered by other authors. The case of constant number of permitted errors was considered in [RMK+80], [Spe84], [Pel87], and [KY90]; disguised as "block coding with noiseless, delayless feedback" the game has appeared as early as 1964 in Elwyn Berlekamp's Thesis [Ber64].

The case where the number of lies is bounded by a fixed fraction of the number of questions, as here, was considered in [Fra90] where the threshold $\frac{1}{2}$ for Version A was found, but Paul's strategy when $r < \frac{1}{2}$ was slow. Pelc [Pel89] obtained a fast win for Paul in Version A when $r < \frac{1}{3}$ but was unable to determine whether Paul can win with $O(\log n)$ questions when $\frac{1}{3} \leq r < \frac{1}{2}$; he left this as his "main open question". Our theorem answers it in the affirmative; Aslam and Dhagat [AD91] have also closed this gap, independently of us, and with a somewhat different proof.

Version B seems not to have been considered before; however, Yossi Azar has pointed out to us that our result for Version B can be obtained also from the analysis in [RMK+80], despite that paper having been directed toward the case of fixed number of lies.

The analysis of Version C turns out, as we shall see, to be an amusing exercise in the theory of error-correcting codes.

The Proof

Versions A and B of our guessing game are conveniently represented by a different, deterministic game involving moving chips on a linear board. In our "chip game" there are n chips, which are marked by the possible answers $1, \ldots, n$. There is a one-dimensional board with positions marked $0, 1, \ldots$. When chip i is at position l it means that if i is Carole's number, that is if $v = i$, then Carole has lied precisely l times.

Initially all chips are on position 0 ("Ask me no questions, and I'll tell you no lies."). On each round Paul first selects a subset A of the chips, corresponding to asking the question "Is $v \in A$?". Now Carole *either*

1. takes all chips not in A and moves them each forward one position,

or

2. takes all chips in A and moves them forward one position.

Option 1 corresponds to a "Yes" answer - for each $i \notin A$ if $v = i$ then Carole has lied one additional time. Similarly, Option 2 corresponds to a "No" answer. In Version A there is a moving gate which after the i-th round is at position $\lfloor ri \rfloor$. The rule then is that any chip that is past the gate is eliminated from the board, never to return. Carole must keep at least one chip on the board, but tries to keep at least two in play since Paul wins the game if at the end of the q rounds there is only one chip remaining.

In Version B the gate is stationary, at position $\lfloor rk \rfloor$, but the game is otherwise the same as in Version A.

Note that in the chip games Carole need not have a value v in mind; she merely endeavors to keep Paul from eliminating $n - 1$ of the n possibilities. Thus the game is deterministic and full-information, and either Paul or Carole must have a winning strategy.

Version A

We shall show first that in Version A, with $r < \frac{1}{2}$, there is a strategy for Paul which determines v with $O(\ln n)$ questions.

Our strategy for Paul will have three stages.

Stage 1: After $O(\ln n)$ rounds only $O(\ln n)$ chips remain.
Stage 2: After a further $O(\ln n)$ rounds only $O(1)$ chips remain.
Stage 3: After a final $O(\ln n)$ rounds only one chip remains.

Stage 1.

There will be $R = c \ln n$ rounds, $c = c_r$ to be determined. In this stage we shall only bother to remove chips after the final round and then if they are in a position l with $l > Rr$. Clearly this change can only help Carole.

Suppose that a fixed number j of rounds remain in Stage 1. We define the *weight* of a chip at this time to be its probability of survival, given that the remainder of its moves in Stage 1 are random. In other words, imagine flipping a fair coin j times and moving the chip forward one square each time a head appears. The chip's weight is the probability that after that process it will lie on a square l with $l \leq rR$.

By a *state* we mean the number of moves remaining, together with the current positions of all the chips. The weight of a state is defined as the sum of the weights of the individual chips. At the start of Stage 1 each chip has weight

$$\Pr[B(R,.5) \leq Rr] \leq 2^{-R(1-H(r))}$$

where $B(n,p)$ is the binomial distribution and $H(x) = -x \lg x - (1-x)\lg(1-x)$ is the entropy function. We select $c = c_r = [(1-H(r))\ln 2]^{-1}$ so that this weight is less than n^{-1} and hence the weight of the initial state is less than one.

Suppose that with j rounds remaining a chip x has a weight denoted $W(x) = W_j(x)$. There will be two possible values for $W_{j-1}(x)$; let us denote by $W^+(x)$ the new (larger) weight if x remains stationary in this round, letting $W^-(x)$ be the weight of x if it moves. The probability definition of weight insures that

$$W(x) = \frac{1}{2}[W^+(x) + W^-(x)].$$

Let S be the state with j rounds remaining. A set A of chips (corresponding to the question "Is $v \in A$?") defines two states S^+, S^-. Here S^- is the state when the chips of A are moved forward one square and S^+ is the state when the chips of the complement of A are moved forward one square. Summing over x, we have that the weight of the state S is the average of the weights of S^+ and S^-. Now we find a choice of A so that the weights of S^+ and S^- lie within one of each other.

To do this we order the chips arbitrarily and we decide sequentially whether or not to place x in A. Let W^+, W^- be the sums over those chips which have been already been placed, so that initially $W^+ - W^- = 0$. Placing x in A adds $W^+(x)$ to W^+ and $W^-(x)$ to W^- so that it adds $W^+(x) - W^-(x)$ to $W^+ - W^-$. Deciding that x is not in A has the reverse effect and subtracts $W^+(x) - W^-(x)$ from $W^+ - W^-$. We always have $0 \leq W^+(x) - W^-(x) \leq 1$, the second inequality following because these weights are probabilities. Place x in A if $W^+ - W^-$ was negative, otherwise leave x out of A. This way the value $W^+ - W^-$ always stays between -1 and 1. The final values of W^+, W^-, the weights of S^+, S^-, are still within one of each other and thus within $\frac{1}{2}$ of the weight of S.

With this strategy, regardless of how Carole plays, the new state at each round has weight at most $\frac{1}{2}$ more than the weight of the old state. Since the initial state had weight less than one, the final state, after $R = c\ln n$ rounds, has weight at most $1 + \frac{R}{2} \sim \frac{c}{2}\ln n$. But the weight after R rounds, with $j = 0$ rounds remaining, is simply the number of chips at positions l with $l \leq Rr$.

Stage 2.

We now restore the moving gate which after the j-th round sits at square $\lfloor rj \rfloor$; recall that any chip which moves past this gate is removed. The gate itself is moving to the right at rate roughly r squares per query. After stage 1 there are $c_1 \ln n$ chips and the gate is at $\sim c_2 \ln n$. Let us move everything $c_1 \ln n$ squares to the right. Letting $c = c_1 + c_2$ the gate is at $c \ln n$ and there are $c_1 \ln n$ chips, all between $c_1 \ln n$ and $c \ln n$. Moving chips to the left (to smaller numbered squares) can only help Carole. (This corresponds to not considering a lie as a lie.) We move the chips so that they all lie between 1 and $c_1 \ln n$ on consecutive distinct squares. We represent this state by the vector 111...1. We thus begin Stage 2 with the gate at $c \ln n$ and $c_1 \ln n$ chips, no two on the same square.

The strategy for Paul is surprisingly simple. Order the chips from left to right, ordering those chips lying on the same square arbitrarily. Paul picks A to be every other chip - i.e. the first, third, fifth, etc. chip. Call a state nice if no two consecutive squares have between them more than two chips. (In particular, no single square can have more than two chips.) We claim that when Paul plays as above from a nice state, the new state must be nice irrespective of whether Carole chooses to move the odd or the even chips. To see this, assume that after the move squares i and $i+1$ together contain more than two chips; let x, y and z be the numbers of chips which stood on squares $i-1$, i and $i+1$ before the move. Then $x>0$ since the extra chip must have come from square $i-1$, while $z<2$ since otherwise the extra chip would have been passed to the right. But $y+z=2$ else there is no problem, so we are reduced to the case $x=y=z=1$; but this also causes no difficulty because then the chip at $i-1$ can only move if the one at $i+1$ moves also. As the initial state is nice Paul insures that all states in Stage 2 are nice.

Since $r<\frac{1}{2}$ we can find an integer $k=k(r)$ so that

$$\frac{k}{2k+1} > \frac{\frac{1}{2}+r}{2}.$$

We claim that with our "alternating" strategy, after $O(\ln n)$ rounds at most $2k$ chips will remain.

Let us define the "progress" of a state to be the sum of the positions (i.e., square numbers) of the $2k+1$ leftmost chips. If ever there are fewer than $2k+1$ chips, Stage 2 terminates. We claim that in each round the progress must increase by at least k.

The subtlety here is that a chip currently among the leftmost $2k+1$ might move past a chip which is not. Since our states are "nice", however, this can only happen if a square is occupied precisely by chips numbered $2k+1$ and $2k+2$ and Carole chooses to move the odd chips. In that case the movement of chip $2k+1$ will not contribute to the increase in progress, but the other k odd chips still do.

The initial weight is nonnegative so after S rounds the weight is at least Sk. After S rounds the gate is at $c\ln n+rS$ so that the progress can be at most $2k+1$ times this quantity. Thus

$$(2k+1)(c\ln n+rS)\geq Sk$$

and so

$$\frac{(2k+1)c}{k-(2k+1)r}\ln n\geq S$$

and Stage 2 terminates in $O(\ln n)$ steps.

Stage 3.

We begin this stage with a fixed number t of chips and the gate at $d_0\ln n$ for some d_0.

Pick any set A of $\lfloor t/2\rfloor$ chips and let Paul select the set A (or whatever remains of it) repeatedly until $|A|$ or more chips go over the gate. If Paul has chosen A for S rounds then at least $|A|$ chips are at least at $\frac{S}{2}$ and the gate is at $c^0\ln n+rS$, so that

$$\frac{S}{2}<c^0\ln n+rS$$

and so

$$S<\frac{c^0}{\frac{1}{2}-r}\ln n$$

Thus $|A|$ chips will be removed before the gate reaches $d_1\ln n$ for some d_1. Applying this strategy $u=\lceil\log_2 t\rceil$ times all the chips but one will be removed before the gate reaches some $d_u\ln n$. The game is now over and the total number of rounds was only $d\ln n$ where d depended only on r.

For Version A it remains only to show that Carole wins when $r = \frac{1}{2}$, provided that there are at least 3 chips at the start. Fortunately this is trivial: at step 1 she can assure that two chips, say chips 1 and 2, remain on the board. Thereafter she moves chips 1 and 2 only when she has to (i.e. when they are split by the query) and on those occasions she moves the leftmost of the two chips. Then chips 1 and 2 move at rate at most $\frac{1}{2}$ and will never cross the gate.

Version B

Let us begin this time with the easy part. Fix $r \geq \frac{1}{3}$ and $n \geq 5$; then we will show that there is *no* number q of questions in which Version B of the game can be won by Paul.

For, suppose otherwise and let $\bar{x}(t) = (x_0, x_1, \ldots, x_n)$ represent the position of the chips at time t, where as usual the chips are renumbered left-to-right at each step, breaking ties arbitrarily. Also, let $k(t)$ be the number of chips still in their initial position (0) at time t. Assume Carole adopts the following strategy:

(1) as long as $k(t) \geq 3$ she makes sure that at each step that no more than $\lfloor k(t)/2 \rfloor$ of the chips move from position 0.

(2) thereafter and for as long as at least three chips are on the board (that is, as long as $x_3 \leq \lfloor rq \rfloor$) she *never allows more than one of the first three chips to move at the same time.*

(3) When only two chips remain, she never moves chip number two.

Let $\phi(t) = x_1 + x_2 + \min(x_3, \lfloor rq \rfloor + 1)$. Then $\phi(t+1) \leq 1 + \phi(t)$ for every $t \geq 1$ until the game ends. Since $n \geq 5$ the first move will leave at least three chips behind, so $\phi(0) = \phi(1) = 0$. When the game ends, say at time t_0, the first chip sits at $\lfloor rq \rfloor$ and the second has just been pushed to $\lfloor rq \rfloor + 1$; hence we have $\phi(t_0) = 3\lfloor rq \rfloor + 2$.

It follows that $q - 1 \geq 3\lfloor rq \rfloor + 2$, but this is impossible since $3\lfloor rq \rfloor \geq 3\lfloor q/3 \rfloor \geq q - 2$.

This concludes the proof that Carole wins Version B when $r = \frac{1}{3}$.

Now we must show that for every $r < \frac{1}{3}$ there is a number M such that Version B of the game can always be won by Paul in at most $M \ln n$ steps.

Here Paul can use the "alternation" strategy throughout, that is, he will at each step number the chips left to right as above, then ask whether x is even. Our task is basically to show that the function ϕ above will *usually* increase at unit rate, when the number of questions is a sufficiently large multiple of $\ln n$.

Let us fix n and q, and temporarily modify the game so that there is no barrier, and so that Carole's objective is simply to minimize the value of $x_1 + x_2 + x_3$ at time q. Paul is using the alternating strategy, so all decision-making is in the province of Carole.

Lemma 1. In this game Carole's best strategy is always to move the even-numbered chips (regardless of the starting position).

Proof of Lemma 1. As before we represent positions by n-tuples $\bar{x} = (x_1, x_2, \ldots, x_n)$ with $x_1 \leq x_2 \leq \ldots \leq x_n$. For $1 \leq k \leq n$ let

$$\sigma_k(\bar{x}) = \sum_{1 \leq i \leq k} x_i$$

so that our objective function is just σ_3. We define the partial order $\leq$ of vectors by $\bar{x} \leq \bar{y}$ if $\sigma_k(\bar{x}) \leq \sigma_k(\bar{y})$ for each k, $1 \leq k \leq n$.

Note that for any position $\bar{x}$, if $\bar{y}$ is the position obtained from $\bar{x}$ either by pushing one chip one step to the right, or by pushing two chips with only emptiness between them toward each other one step each, then $\bar{x} \leq \bar{y}$. Conversely, if $\bar{x} \leq \bar{y}$ but $\bar{x} \neq \bar{y}$, and we let k be the maximal subject to $\sigma_k(x) < \sigma_k(y)$, then moving chip k of the $\bar{x}$ configuration one step to the right and chip $k+1$ (if $k \neq n$) one step to the left results in a position $\bar{x}'$ such that $\bar{x} < \bar{x}' \leq \bar{y}$. Hence a finite sequence of such operations will convert $\bar{x}$ to $\bar{y}$.

Let $e(\bar{x})$ denote the position achieved by moving the even-numbered chips from position $\bar{x}$, and $o(\bar{x})$ the result of moving the odd chips from $\bar{x}$. We claim first that for any $\bar{x}$, $e(\bar{x}) \leq o(\bar{x})$; this immediately proves the lemma for the special case $q = 1$.

To verify the claim, number the chips in accordance with their positions in $\bar{x}$ and observe that if for each odd i chips i and $i+1$ are moved toward each other, then $e(\bar{x})$ is converted to $o(\bar{x})$. In some cases chips may switch positions (to no effect) and the nth chip will move to the right by itself if n is odd, but we have in any case that $e(\bar{x}) \leq o(\bar{x})$.

Next, we claim that if $\bar{x} \leq \bar{y}$ then $e(\bar{x}) \leq e(\bar{y})$. Here we may assume that $\bar{y}$ is obtained from $\bar{x}$ by a single operation; if by pushing one chip to the right, the result is obvious. Otherwise we may take the effected chips to be those numbered k and $k+1$ in $\bar{x}$, where $x_{k+1} - x_k \geq 2$; it is now again easily checked that $e(\bar{x}) \leq e(\bar{y})$. Note however that if k is even and $x_{k+1} - x_k = 2$ then we get $e(\bar{x}) = e(\bar{y})$ even though $\bar{x}$ and $\bar{y}$ are distinct.

Now for any q and $\bar{x}$, let $\psi_q(\bar{x})$ be the least possible value of $\sigma_3(q)$ given position $\bar{x}$ at time 0. Then ψ satisfies the recursion $\psi_q(\bar{x})=\min(\psi_{q-1}(e(\bar{x})),\ \psi_{q-1}(o(\bar{x})))$.

We see by induction on q that $\bar{x} \leq \bar{y}$ implies $\psi_q(\bar{x}) \leq \psi_q(\bar{y})$, for $\bar{x} \leq \bar{y}$ implies $e(\bar{x}) \leq e(\bar{y})$ which forces $\psi_{q-1}(e(\bar{x})) \leq \psi_{q-1}(e(\bar{y})) \leq \psi_{q-1}(o(\bar{y}))$ and $\psi_{q-1}(e(\bar{x})) \leq \psi_{q-1}(o(\bar{x}))$ by the induction hypothesis. Thus $\psi_q(\bar{x}) = \psi_{q-1}(e(\bar{x})) \leq \psi_{q-1}(e(\bar{y})) = \psi_q(\bar{y})$ as required.

As a result we have that always

$$\min(\psi_{q-1}(e(\bar{x})), \psi_{q-1}(o(\bar{x}))) = \psi_{q-1}(e(\bar{x}));$$

this means that Carole need never select "o", proving the lemma. ∎

Our next task is to obtain a lower bound for $\psi_q(\bar{0})$; since we know Carole's best strategy we may assume that she uses it. In that case chip number 1 (i.e. the chip sitting in position 0 when the initial stack reduces to one) never moves, and if n is even, chip number n moves every time. We will temporarily trim off those one or two chips, thus reducing to the case where n is even and Carole moves the *odd* chips at every step.

These chips live in a slowly broadening "clump" which moves at rate $\frac{1}{2}$. A more general discussion of the movement of this clump is given in the final section. Let $f(n,t)$ denote the position of the leftmost chip (i.e., the chip with the minimal position) at time t, beginning with n chips at time zero.

Lemma 2. For all $\delta > 0$ there exists K so that if $t > K \ln n$ then

$$f(n,t) > \frac{t}{2}(1-\delta).$$

Remark. Each time unit the average position of the n chips increases by $\frac{1}{2}$, since precisely $\frac{1}{2}$ of the chips move forward one. Thus $t/2$ is the "center of gravity" of the n chips and Lemma 2 says that they do not spread out too rapidly. This is close to best possible in the sense that if $t = \lfloor \lg n \rfloor$ then at time t there is still a chip on square zero.

To prove Lemma 2 we appeal to yet another lemma.

Lemma 3. The chip position at time t is symmetric about $t/2$.

Proof of Lemma 3. We proceed by induction on t. Assume true for t and order the chip positions $p_1 \le p_2 \le \ldots \le p_n$ so that $p_i + p_{n+1-i} = t$. Exactly one of $i, n+1-i$ are odd so exactly one of these two chips move and so these two chips are symmetric about $(t+1)/2$ at time $t+1$. The n chips split into $n/2$ pairs, each symmetric about $(t+1)/2$, hence the position is symmetric.

We now return to the proof of Lemma 2. Let $p_1 \le p_2 \le \ldots \le p_n$ denote the positions at time t and set

$$F_t = \sum_{1 \le i \le n} (1+\epsilon)^{2\left|p_i - \frac{t}{2}\right|}$$

where $\epsilon \in (0, .1)$ is, for the moment, fixed but arbitrary. We claim that

$$F_{t+1} \le F_t e^{\epsilon^2/2} + n\epsilon$$

Let p_i^* denote the position at time $t+1$ of what was the i-th chip at time t. (Of course, that chip may have some different ordinal at time $t+1$.) For $1 \le i \le n$ define

$$w_i = (1+\epsilon)^{2\left|p_i - \frac{t}{2}\right|}$$

and

$$w_i^* = (1+\epsilon)^{2\left|p_i^* - \frac{t+1}{2}\right|}$$

so that $F_t = \sum w_i$ and $F_{t+1} = \sum w_i^*$.

First consider a pair of chips with ordinals (at time t) $2i-1, 2i$ with $2i \le n/2$ and with $p_{2i-1} < t/2$. In this case $p_{2i-1}^* = p_{2i-1} + 1$, the absolute value in the exponent for both w_{2i-1} and w_{2i-1}^* are in the same direction, hence

$$w_{2i-1}^* = w_{2i-1}/(1+\epsilon)$$

That is, the chip has moved one half closer to the mean. The chip $2i$ stands still and so is one half further from the mean and

$$w_{2i}^* = w_{2i}(1+\epsilon)$$

The weight function w is monotone in the distance from the mean so since $p_{2i-1} \le p_{2i}$, $w_{2i-1} \le w_{2i}$ and therefore

$$w_{2i-1}^* + w_{2i}^* \le (w_{2i-1} + w_{2i}) \frac{(1+\epsilon) + (1+\epsilon)^{-1}}{2}$$

That is, the heavier weighted chip is losing weight so that the proportion of weight gained is at most if the weights were equal. We simplify by the inequality

$$\frac{(1+\epsilon) + (1+\epsilon)^{-1}}{2} \le 1 + \frac{\epsilon^2}{2} \le e^{\epsilon^2/2}$$

so that

$$w_{2i-1}^* + w_{2i}^* \le (w_{2i-1} + w_{2i}) e^{\epsilon^2/2}$$

The same inequality holds (for symmetric reasons) when $2i - 1 \ge 1 + \frac{n}{2}$ and $p_{2i} > t/2$. The only time the inequality might not hold is when t is even and $p_{2i-1} = p_{2i} = t$. In that event both chips were *on* the mean and both are now $\frac{1}{2}$ from the mean. (The culprit is the absolute value sign in the weight function.) In that case $w_{2i-1} = w_{2i} = 1$ and

$$w^*_{2i-1} + w^*_{2i} \le (w_{2i-1} + w_{2i}) + 2\epsilon$$

Thus for *all* i

$$w^*_{2i-1} + w^*_{2i} \le (w_{2i-1} + w_{2i})e^{\epsilon^2/2} + 2\epsilon$$

and summing over i gives the Claim. We have that

$$F_t \le F_{t-1}e^{\epsilon^2/2} + n\epsilon$$

$$F_0 = n$$

Standard methods give

$$F_t \le ne^{t\epsilon^2/2}c_\epsilon$$

where

$$c_\epsilon = 1 + \frac{\epsilon}{1 - e^{-\epsilon^2/2}} < \frac{3}{\epsilon}$$

for, say, $\epsilon < .1$.

Now let $A = 2(\frac{t}{2} - f(n,t))$. At time t the chip at position $f(n,t)$ contributes $(1+\epsilon)^A$ to the weight F_t so that

$$(1+\epsilon)^A \le F_t \le \frac{3}{\epsilon}ne^{t\epsilon^2/2}$$

and so, taking logarithms,

$$A \le \frac{\ln n}{\epsilon} + \frac{t\epsilon}{2} + \frac{\ln(2/\epsilon)}{\epsilon}$$

Ignoring the third term we optimize by setting

$$\epsilon = \sqrt{\frac{2\ln n}{t}}$$

(The formal proof begins by setting ϵ equal this value.) It is convenient to reparameterize $t = L(2\ln n)$. Then $\epsilon = L^{-1/2}$. For K sufficiently large, $L \ge 100$ and $\epsilon \le .1$ as we assumed. For this ϵ

$$A \le \frac{t}{2\sqrt{L}} + \frac{t}{2\sqrt{L}} + \sqrt{L}\ln(2\sqrt{L}) \le t\left[\frac{1}{\sqrt{L}} + \frac{\ln(2\sqrt{L})}{2\ln 2\sqrt{L}}\right]$$

as $n \ge 2$. Now pick K so that for $L \ge K/2$ the bracketed term is less than $\delta/2$. ∎

This concludes the proof of Lemma 2, but let us remark on its implications for $\psi(\bar{0})$. Recall that in switching from the "even" to the "odd" Carole strategy, one chip was permanently left behind in its initial position. The second and third chips are past $(1-\delta)t/2$ for $t > K\ln n$, so that $\psi(\bar{0}) > (1-\delta)t$. It follows that our original function $\phi(t)$ has also exceeded $(1-\delta)t$, provided there are still at least three chips on the board at time t.

Now let us return to the original chip game (Version B) with Paul playing the alternating strategy. Let $\delta = \frac{1}{3} - r$ and choose K as in Lemma 2. Let $M = K/\delta$ and fix $q = \lceil M\ln n\rceil$; we will show that q questions suffice for Paul to win.

Assume otherwise; then Carole has a winning strategy which will find at least two chips still on the board after q steps. Let $t_1 \le q$ be the last time at which there are at least *three* chips on the board.

Between times t_1 and q, ϕ must increase at precisely unit rate; and thus $\phi(t_1) \le 3\lfloor rq \rfloor + 1 - (q - t_1)$. If $t_1 \le K \ln n$ then

$$0 \le \phi(t_1) \le 3(\frac{1}{3} - \delta)M \ln n - (M - K)\ln n$$
$$\le M \ln n((1 - 3\delta) - (1 - \delta)) < 0,$$

a contradiction.

On the other hand if $t_1 > K \ln n$ then

$$(1 - \delta)t_1 < \phi(t_1) \le (1 - 3\delta)q - (q - t_1)$$

which implies $t_1 > 3q$, the final contradiction.

Version C

We now proceed to Version C, in which the questions are submitted to Carole in a "batch" and she can look them all over before selecting her answers. Recall that here we need to establish that Carole has a fast win when $r < \frac{1}{4}$, a slow win when $r = \frac{1}{4}$, and no win when $r > \frac{1}{4}$.

In version C the chip model is no longer useful. Instead, we may imagine that Paul submits to Carole a q-by-n {-1,1}-matrix M such that the ith query becomes "is $M_{i,x} = 1$?" where as before x stands for the "true" value.

Carole's response is a q-vector A of answers where $A_i = 1$ means "yes" and $A_i = -1$ means "no". Thus, the number of lies issued when $x = j$ is precisely the number of -1's in the jth column of the matrix M' obtained by multiplying row i of M by A_i, for each i.

Lemma 4. Let r, n and q be fixed. Then a q-by-n matrix M wins for Paul if and only if the Hamming distance $d(C_j, C_{j'})$ between any two distinct columns of M is strictly greater than $2\lfloor rq \rfloor$.

Proof of Lemma 4. Note first that the Hamming distances between columns of M' are the same as in M, and that Carole wins just if she can arrange to have at least two columns of M' of distance at most $\lfloor rq \rfloor$ to the q-vector (1,1,...,1); but this is impossible when the condition of Lemma 4 holds, on account of the triangle inequality.

On the other hand, if there are two columns i, i' of distance no more than $2\lfloor rq \rfloor$ then there is a vector V at distance no more than $\lfloor rq \rfloor$ from each of them; by choosing V as her answer vector Carole keeps both i and i' as possible values of x. ∎

We see from Lemma 4 that Version C of our game is essentially a problem in the theory of error-correcting codes. Examples of the techniques used in the proof of the following theorem can be found in [vL82].

We now proceed to the case where $r < \frac{1}{4}$. It turns out that it suffices for Paul to use any "greedy" method for constructing his matrix; that is, he selects each new column so that its Hamming distance to previous columns is sufficiently high. The following deterministic method is perhaps the cleanest, but the proof can be applied to any greedy construction.

Let us now take

$$q = \lceil (\log_2 n)/(1 - H(2r)) \rceil,$$

where H is the binary entropy function given by

$$H(\lambda) = -\lambda \log_2 \lambda - (1-\lambda)\log_2(1-\lambda).$$

We order the vectors $\bar{x} = (x_1, x_2, \ldots, x_q)$ lexicographically—so that, say, $\bar{x} < \bar{y}$ iff $x_i = -1$ and $y_i = 1$ where i is the least index such that $x_i \neq y_i$.

Now Paul constructs M column by column, always choosing the lexicographically least vector which is at Hamming distance at least $1 + 2\lfloor rq \rfloor$ from all previously chosen columns.

To see that this works, note that there are fewer than $2^{qH(2r)}$ vectors at distance no more than $2rq$ from a given vector, hence Paul cannot get stuck choosing columns as long as $n \leq 2^q/2^{qH(2r)} = 2^{(1-H(2r))q}$; but this inequality follows by choice of q.

Interestingly, if n is a power of 2 then the columns form a linear subspace. The resulting linear code is called a "lexicode" by John Conway and has many surprising properties.

Let us look next at the case where $r = \frac{1}{4}$ exactly. We show first that Paul cannot win with fewer than $n/4 + 1$ questions.

To see this let M be a winning q-by-n matrix. At least $n/4$ of the columns of M must share the same last two entries; let M' be formed from these columns, with the last two entries of each column cut off. Then M' is a $(q-2)$-by-m matrix, where $m \geq n/4$, and still has minimum distance greater than $2\lfloor q/4 \rfloor$.

For M' each row can contribute no more than $m^2/4$ to the total distance between columns, which is at least $(1 + 2\lfloor q/4 \rfloor)m(m-1)/2$. The critical case is when $q = 4k+3$, where we have

$$(4k + 3 - 2)m^2/4 \geq (1 + 2k)m(m-1)/2$$

which implies that $m \leq 4k + 2 = q - 1$, thus $n/4 + 1 \leq q$ as required.

On the other hand, let H be a Hadamard matrix, that is, an h-by-h matrix of 1's and -1's with $HH^t = hI$. Such matrices exist for many values of h which are multiples of 4, e.g. $h = 1 + p^r$ for an odd prime p. (See [Hal86].)

Let us form M by placing a copy of H^t and a copy of $-H^t$ side by side, then truncating the last row. Then M will be a $h-1$-by-$2h$ matrix whose minimum distance between columns is at least $h/2 - 1 = 1 + 2\lfloor h - \frac{1}{4} \rfloor$. For $n = 2h$ this will win the game for Paul with $n/2 - 1$=O(n) questions.

Suppose now that $r > \frac{1}{4}$, and let $n \geq 9r/(r - \frac{1}{4})$. We claim that there is no matrix M that will win for Paul.

Arguing as above, if there were such an M, then the sum of the distances between pairs of columns of M would be at least

$$(1 + 2\lfloor rq \rfloor)n(n-1)/2 > (-1 + 2rq)n(n-1)/2;$$

but each row of M can contribute no more than $n^2/4$ to this total. Thus

$$(-1 + 2rq)n(n-1)/2 < qn^2/4$$

and solving for n, we obtain

$$n < \frac{r - 1/2q}{r - \frac{1}{4} - 1/2q}.$$

However, we may still use the lower bound $q \geq n/4 + 1$ from the $r = \frac{1}{4}$ case above, whence

$$1/2q < 2/n \leq (r - \frac{1}{4})/(9r/2) \leq \frac{8}{9}(r - \frac{1}{4}).$$

It follows that

$$n < \frac{r}{(r - \frac{1}{4}) - \frac{8}{9}(r - \frac{1}{4})} = \frac{9r}{r - \frac{1}{4}}$$

contradicting the choice of n.

This, finally, concludes the proof of the Theorem. ∎

Acknowledgment: The authors have benefitted from conversations with Aditi Dhagat and Prasad Tetali.

References

[AD91] Javed Aslam and Aditi Dhagat. Searching in the presence of linearly bounded errors. to appear in STOC 1991, 1991.

[Ber64] E. Berlekamp. *Block Coding with Noiseless Feedback.* PhD thesis, M.I.T., 1964.

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

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

[Hal86] Marshall Hall. *Combinatorial Theory (2nd ed.).* Wiley, 1986.

[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.

[Pel89] Andrzej Pelc. Searching wih known error probability. *Theoretical Computer Science*, 63:185–202, 1989.

[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.

[Spe84] Joel Spencer. Guess a number - with lying. *Mathematics Magazine*, 57(2):105–108, 1984.

[Spe86] Joel Spencer. Balancing vectors in the max norm. *Combinatorica*, 6:55–65, 1986.

[Ula76] S.M. Ulam. *Adventures of a Mathematician.* Charles Scribner's Sons, 1976.

[vL82] J.H. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1982.