International Journal of Mathematics
and Computer Science, 1(2006), 37-61

Average-case Analysis of the Chip Problem

Laurent Alonso[1], Philippe Chassaing[2], Edward M. Reingold[3] & René Schott[2]

[1] INRIA-Lorraine and LORIA, Université Henri Poincaré-Nancy I
BP 239, 54506 Vandoeuvre-lès-Nancy, France
e-mail: Laurent.Alonso@loria.fr

[2] Institut Élie Cartan, Université Henri Poincaré-Nancy I
BP 239, 54506 Vandœuvre-lès-Nancy, France
e-mail: Philippe.Chassaing@antares.iecn.u-nancy.fr; Rene.Schott@loria.fr

[3] Department of Computer Science, Illinois Institute of Technology
Stuart Bldg, 10 W. 31st St., Suite 236, Chicago, Illinois 60616-2987 USA
e-mail: reingold@iit.edu

(Received August 2, 2005, Accepted September 19, 2005)

Abstract

In the system level, *adaptive fault diagnosis problem* we must determine which components (chips) in a system are defective, assuming the majority of them are good. Chips are tested as follows: Take two chips, say x and y, and have x report whether y is good or bad. If x is good, the answer is correct, but if x is bad, the answer is unreliable and can be either wrong with probability α or right with probability $1-\alpha$. The key to identifying all defective chips is to identify a single good chip which can then be used to diagnose the other chips; the *chip problem* is to identify a single good chip. In [1] we have shown that the chip problem is closely related to a modified majority problem in the *worst case* and have used this fact to obtain upper and lower bounds on algorithms for the chip problem. In this paper, we show the limits of this relationship by showing that algorithms for the chip problem can violate lower bounds on *average performance* for the modified majority problem and we give an algorithm for the "biased chip" problem (a chip is bad with probability p) whose average performance is better than the average cost of the best algorithm for the biased majority problem.

1 Introduction

In system diagnosis, according to [14], we consider

> a system consisting of a set U of n units (processors, modules, etc.) at most t of which are faulty. An external observer wishes to identify the faulty units. The observer acquires information by requesting the results of certain tests performed by one unit upon another; e.g.,

Key words and phrases: Fault diagnosis, algorithm analysis, chip problem, majority problem, probabilistic algorithms.
AMS (MOS) Subject Classifications: 68Q25, 68P10, 68Q20, 68M15
ISSN 1814-0424
This research was supported in part by INRIA and the NSF, through grant numbers NSF INT 90-16958 and 95-07248. Supported in part by NSF grants CCR-93-20577 and CCR-95-30297 and by a Meyerhoff Visiting Professorship at the Weizmann Institute of Science

> $u_i \in U$ might be asked to determine if $u_j \in U$ is faulty or not... If u_i is fault-free then [the test performed by u_i] is assumed reliable; if u_i is faulty however, u_i may find u_j either faulty or fault-free, regardless of the actual condition of u_j.

For example, suppose we have a combination of electronic components in an inaccessible location (in outer space, under the sea, in a deadly environment such as a nuclear reactor, and so on). We are able to test components only at great expense, and only relative to other components—when we get test results, the testing components themselves may be faulty. We thus have a problem in which we get results such as "According to component x, component y is functioning properly" or "According to component x, component y is defective." It is a complex issue to determine which components are defective under such conditions; furthermore, since communication itself is difficult, we do not want to expend unnecessary effort by asking more questions than the minimum number needed.

For convenience of language we follow [7, exercise 4-7, page 75] and refer to "chips" rather than "units" or "components". An algorithm can determine whether a bad (faulty) chip exists if and only if a strict majority of the chips are fault-free [12]. The basic problem is to determine that some chip is good (fault-free) so we can rely on its diagnosis of other chips; the difficulty is that a faulty chip can behave exactly like a fault-free one. However, there are configurations of test results in which the assumption that some particular chip x is bad implies that a majority of chips are faulty too—since we know that a strict majority are good, we are then sure that x is good, and we can rely on its diagnosis; Figure 1 shows just such a configuration. There are a number of attendant algorithmic questions, including designing and analyzing algorithms and determining lower bounds for a variety of problems. These problems have a long history and lengthy bibliography; Pelc and Upfal [11] give an excellent summary with a useful bibliography. In this paper we address only the *chip problem*—the question of finding a single good chip, assuming that the majority of the chips are good and that any chip can test any other. We consider both the worst and average case for this problem, using results of the "majority problem" as starting point of our investigations.

This paper is organized as follows. We begin with a brief section introducing the necessary background of the majority problem. Then, in Section 4, we give results and conjectures about the average case under two different sets of probablistic assumptions. We conclude in Section 5 with some observations and open problems.

2 Majority Problems

The *majority problem* of [2], [3], and [13] is to determine the majority color in a set of n elements $\{x_1, x_2, \ldots, x_n\}$, each element of which is colored either blue or red, by pairwise equal/not equal color comparisons. When n is even, we must

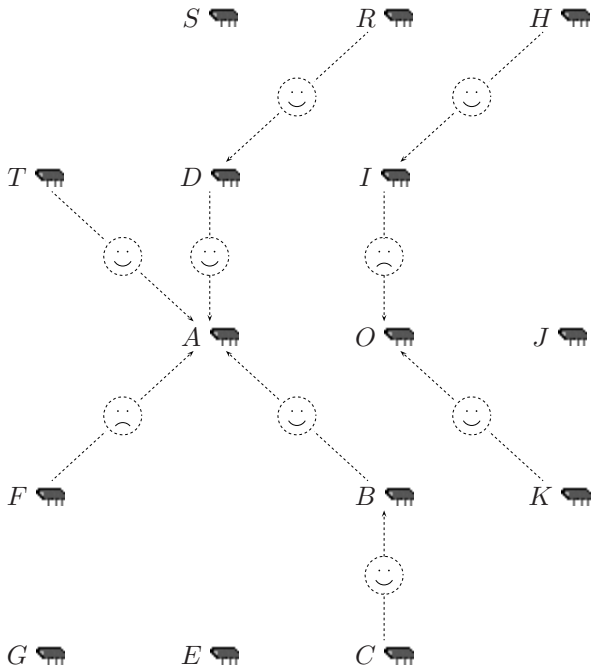

Figure 1: Sample test configuration of fifteen chips. We show a link from one chip to another labeled by a smiling face if the chip at the base of the arrow diagnoses the chip at the point of the arrow as good, and label the link with a frowning face if the diagnosis is bad. Chip F must be bad and chip A must be good by the following logic: If A were bad, chips D, T, and B would be bad because each of them diagnoses A as good; then, similarly, chips R and C would have to be bad, making the set $\{A, B, C, D, R, T\}$ of six chips all bad. However, since I diagnoses O as bad, one of those two must be bad—O is bad if I is good while I is bad if O is good—implying that one of H or K must also be bad. Thus if A were bad there would be at least eight bad chips, a majority of the chips.

report that there is no majority if there are equal numbers of each color. In the worst case, exactly

$$n - \nu(n)$$

questions are necessary and sufficient for the majority problem, where, following [8], $\nu(n)$ is the number of 1-bits in the binary representation of n. This result was first proved by Saks and Werman [13]; [2] gave a short, elementary proof; [17] found yet a different approach. [3] proved that any algorithm that correctly determines the majority must on the average use at least

$$\frac{2n}{3} - \sqrt{\frac{8n}{9\pi}} + \Theta(1)$$

color comparisons, assuming all 2^n distinct colorings of the n elements are equally probable. Furthermore, [3] describes an algorithm that uses an average of

$$\frac{2n}{3} - \sqrt{\frac{8n}{9\pi}} + O(\log n)$$

color comparisons. Together these bounds imply that

$$\frac{2n}{3} - \sqrt{\frac{8n}{9\pi}} + O(\log n)$$

such comparisons are necessary and sufficient in the *average case* to solve the original majority problem.

The biased case of the majority problem has been investigated by Chassaing [6]. Let p be the probability that an element is blue and $q = 1 - p$ the probability that it is red. Let $\rho = q/p$. Chassaing proved that

$$\Phi(\rho)n + O(n^{\frac{1}{2}+\epsilon})$$

comparisons are necessary and sufficient in the biased case of the majority problem on n elements, where

$$\Phi(\rho) = \frac{1-\rho}{4} \sum_{i=0}^{\infty} \frac{1+\rho^{2^i}}{2^i(1-\rho^{2^i})}. \tag{1}$$

Note that the function Φ is well defined for $\rho = 1$ since $\lim_{\rho\to 1} \Phi(1) = 2/3$; note further that $\Phi(0) = 1/2$. Tedious computation shows that, as expected, Φ is increasing in ρ.

In the *modified majority problem*, we are guaranteed the existence of a strict majority. For odd n, the original and modified problems are obviously identical and bounds of the previous paragraphs apply directly to the modified majority problem. But when n is even, the modified problem is clearly no harder than the original problem for $n - 1$ elements and may even be (a bit) simpler. Given algorithm M for the $(2k-1)$-original-majority-problem, we solve the $(2k)$-modified-majority-problem by removing one element and running algorithm M on the $2k-1$ remaining elements—the answer is then obviously also

correct for the $(2k)$-modified-majority-problem. The other direction is open: Given algorithm M' for the $(2k)$-modified-majority-problem, can we apply it to the $(2k-1)$-original-majority-problem? If so it would prove that the modified problem for $2k$ elements is no harder than the original problem for $2k-1$ elements and establish the worst-case lower bound of $2k-1-\nu(2k-1)$ color comparisons. We believe this lower bound is correct, but cannot prove it.

3 Happy Trees

The key to identifying a good chip is to build trees of chips such that if the chip at the root is bad, then all chips in the tree must be bad; when a sufficiently large such tree has been built, the condition that a strict majority of the chips is good guarantees that the root chip must be good. The tree structures we use are *binomial trees* [16] (see also [7, Chapter 20]). For convenience, we summarize the necessary details here.

A binomial tree of chips $\mathcal{B}_k$ is an (unordered) tree defined recursively as shown in Figure 3. $\mathcal{B}_0$ consists of a single chip and $\mathcal{B}_k$ consists of two binomial trees of chips $\mathcal{B}_{k-1}$ in which the root of one has tested the root of the other. We show a link from child to parent labeled by a smiling face if the child chip says the parent chip is good, and a link from parent to child labeled with a frowning face if the parent chip says the child is bad. We call a binomial tree *happy* if it has only smiling faces on its links. The *order* of the binomial tree $\mathcal{B}_k$ is k.

Suppose the chip r_1 at the root of one happy binomial tree tests the chip at the r_2 root of a second happy binomial tree and says r_2 is good. Then (inductively), if r_2 is bad, all chips in both binomial trees must be bad, so when we combine the two trees by linking r_1 as a child of r_2, we get a happy binomial tree in which if the root chip is bad, all chips in the tree must be bad. But, if the chip r_1 at the root of one binomial tree tests the chip at the r_2 root of a second binomial tree and says r_2 is bad, then if r_2 *is* bad, all chips in its tree are bad, while if r_2 is good, r_1 must be bad and all chips in its tree are bad; in either case, at least half of the chips in the union of the two trees must be bad and they can be ignored when seeking a good chip.

In some algorithms for the chip problem we deviate from binomial trees and combine two trees of different sizes. In such cases, as we will see, the difference between the two sizes constitutes a bound on the plurality between good chips and bad chips in the trees.

4 The Chip Problem in the Average Case

In [1], we showed that there is a close relationship between algorithms for the chip problem and those for the modified majority problem. In this section we show the limits of this relationship by showing that algorithms for the chip problem can violate lower bounds on average performance for the modified majority problem.

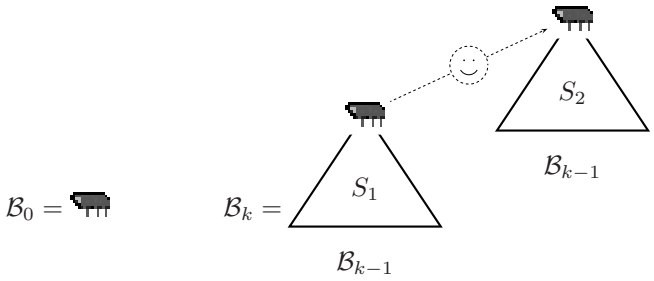

Figure 2: Binomial trees of chips. The smiling face indicates the chip at the base of the arrow says that the chip at the point of the arrow is good. If all arrows in the trees S_1 and S_2 are smiling and the chip at the root of S_2 is bad, all chips in both S_1 and S_2 must be bad by induction. Since $\mathcal{B}_0$ contains a single chip, $\mathcal{B}_k$ contains 2^k chips.

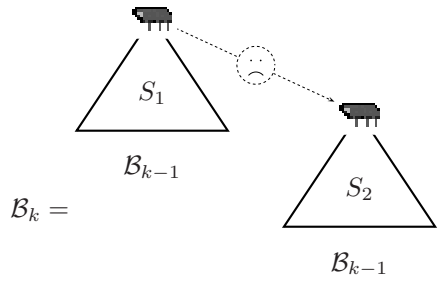

Figure 3: The frowning face indicates the chip at the base of the arrow says that the chip at the point of the arrow is bad. Suppose all the arrows in the trees S_1 and S_2 are smiling. If the chip at the root of S_1 is good, all chips in S_2 must be bad; if the chip at the root of S_1 is bad, all chips in S_1 must be bad. In either case, at least half the chips in $S_1 \cup S_2$ must be bad and since $|S_1| = |S_2| = 2^{k-1}$, $S_1 \cup S_2$ can be ignored in the search for a good chip.

4.1 The Model and Previous Results

We use a variant of the model of Pelc and Upfal [11], who consider that each chip is good independently with known probability p and bad with probability $q = 1-p < p$; the number N of good chips thus has a binomial distribution. Let Q be the probability that an adaptive diagnosis algorithm incorrectly identifies the set of faulty chips. Pelc and Upfal prove that the minimal number of tests required to identify all faulty chips in the worst case is $n + \Theta(\log \frac{1}{Q})$. There are three main differences between their results and ours: we limit ourselves to algorithms *that never fail*; we study the *average* number of tests required; and we focus on the problem of finding a single good chip, assuming at most $n-1$ additional tests will be needed to determine the status of the $n-1$ other chips. According to [12], a configuration with $N \leq n/2$ good chips can fool any algorithm, but there do exist algorithms that reliably diagnose any configuration with a strict majority of good chips. Thus in order for reliable algorithms to exist, we must preclude the possibility that $N \leq n/2$, so we consider the model of Pelc and Upfal with the additional condition that $N > n/2$; that is, we assume that each partition of the chips into $k > n/2$ good chips and $n-k$ bad chips has probability

$$\frac{p^k q^{n-k}}{\mathbf{Pr}(N > n/2)}. \tag{2}$$

As we show in Proposition 2, this is only a slight change, since we have, for $p > q$,

$$\begin{aligned} \mathbf{Pr}(N \leq n/2) &= \sum_{0 \leq k \leq n/2} \binom{n}{k} p^k q^{n-k} \\ &\leq e^{-(p-q)^2 n/2}, \end{aligned} \tag{3}$$

and hence the denominator of (2) is close to 1.

The computation of the average cost of algorithms requires a probabilistic model for diagnosis of faulty chips; we use a simple instance of those surveyed in [10]: given a bad chip x and some other chip y, we assume that x lies about y with probability α, and tells the truth with probability $1-\alpha$. We assume that p and α are known, but this in no way weakens the results since in the likely case that p and α are unknown, we could obtain good approximations of them at cost $o(n)$, using statistics on a sample of $n^{1-\varepsilon}$ chips.

Note that if we know that $\alpha = 0$, no chip lies, and the problem has trivially average cost $O(1)$. However, in the setting where p and α are unknown but where $\alpha = 0$, we will find any no erroneous diagnoses in our sample of $n^{1-\varepsilon}$ chips; this would only tell us that α is quite small, *not* that $\alpha = 0$. So, we could not assume that each diagnosis is reliable, and we could only rely on a majority of chips being good; thus not knowing that $\alpha = 0$ leads to a lower bound $\Theta(n)$ for the best average cost, *not* $O(1)$.

Choosing $Q = \mathbf{Pr}(N \leq n/2)$ in Pelc and Upfal [11] allows comparison of their results to ours. Using (3), their result translated to the problem of finding

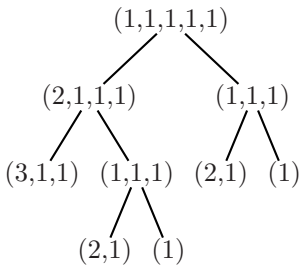

Figure 4: Optimal tree for $n = 5$ in the majority problem [3]. It has expected cost 2.25.

a single good chip gives the trivial worst case bound of $\Theta(\log \frac{1}{Q}) = \Theta(n)$. It is equally easy to see that the minimal average cost is $\Theta(n)$. The problem we address is much more difficult: determining the value of the multiplicative constant in $\Theta(n)$. Thus we have to design the algorithm more carefully than in [11].

4.2 The Unbiased Case

In the simple *unbiased case*, we ignore p and q and assume instead that a set of n good and bad chips is chosen at random so that each set of chips with a majority of good chips is equally probable. That is, each of the $\sum_{n/2<k\leq n} \binom{n}{k}$ sets of n chips having a strict majority of good chips occurs with probability $1/\sum_{n/2<k\leq n} \binom{n}{k}$.
Since the modified majority problem is identical to the original majority problem for odd n, the tree in Figure 4 (which is derived from the algorithm described in the conclusions of [3]), is optimal for the modified majority problem with $n = 5$, having cost 2.25. On the other hand, Figure 5 shows the identical tree for the chip problem with $n = 5$ having expected cost $2.25 - (\alpha - \alpha^2)/16$ if each of the 16 possible configurations is equally probable. Since $(\alpha - \alpha^2)/16 > 0$ for $0 < \alpha < 1$, this proves that the chip problem and the modified majority problem are *not* the same in the average case for $0 < \alpha < 1$. An even lower expected cost of $2 + \alpha(5 - \alpha)/16$ is obtained by having chip 3 test chip 2 in the left child of the root in Figure 5; exhaustive consideration of all possible trees shows that such a tree is the optimal way to solve the chip problem for $n = 5$. This example shows that the unbiased case of the chip problem is indeed different in average behavior from the majority problem, despite the similarity of these problems in the worst case[1].

4.3 The Biased Case

Set

$$\rho = \frac{q}{p};$$

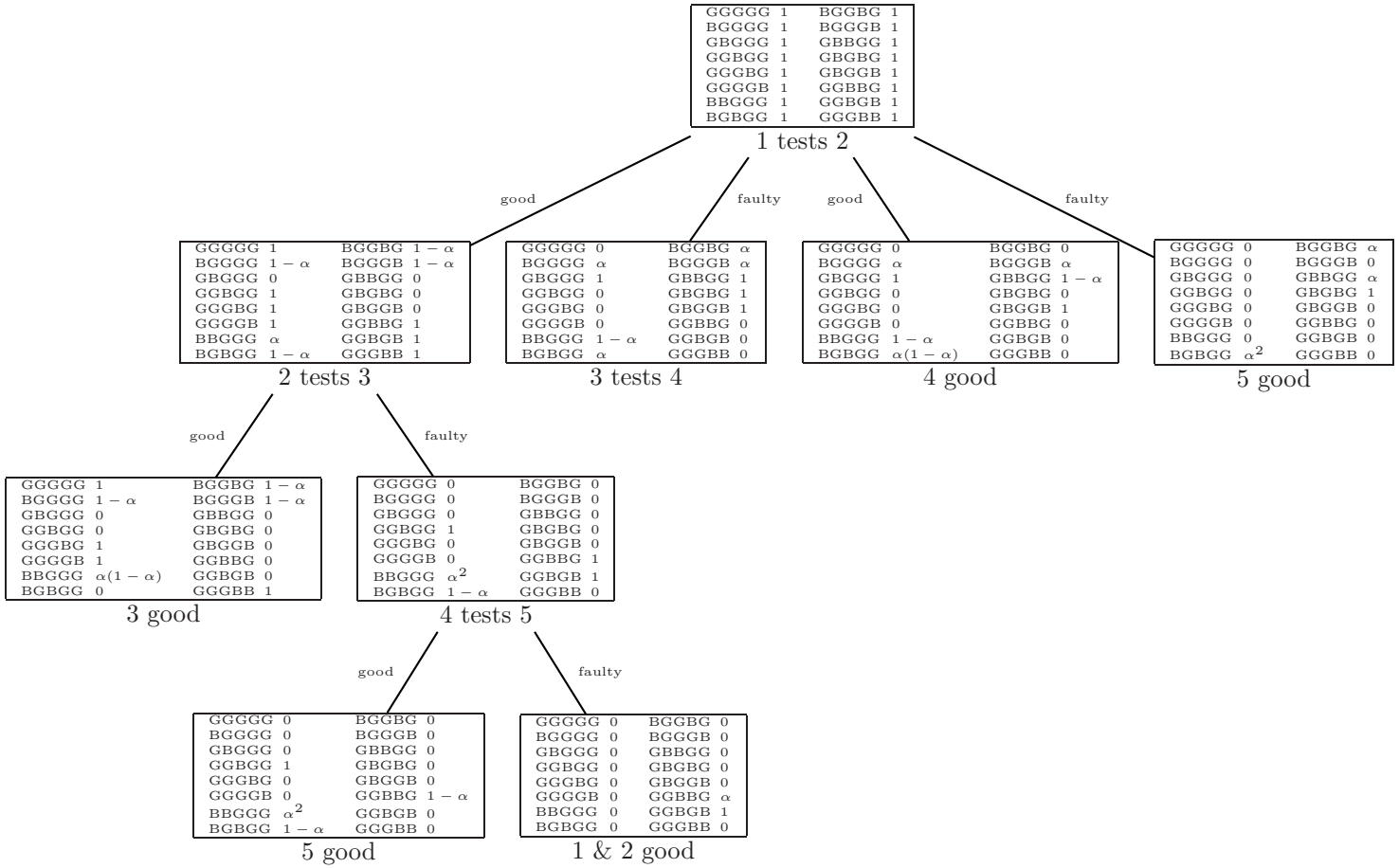

Figure 5: Tree (derived from that in Figure 4) for $n = 5$ in the chip problem; in the unbiased case (if each of the 16 possible configurations is equally probable) its expected cost is $2.25-(\alpha-\alpha^2)/16$. Each configuration is shown along with its frequency of occurrence, assuming a faulty chip gives a wrong answer with probability α. An even lower expected cost of $2+\alpha(5-\alpha)/16$ is obtained by having 3 test 2 in the left child of the root—exhaustive consideration of all possible trees shows that such a tree is optimal.

since $q < p$, $\rho < 1$. Let $C_n(\rho, \alpha)$ be the average-case complexity for the n-chip problem; that is, $C_n(\rho, \alpha)$ is the average number of tests that are necessary and sufficient to identify a good chip in the biased case. In this section, we give an upper bound on $C_n(\rho, \alpha)$ by describing a family $A_n(\rho, \alpha, \epsilon)$, $n \geq 1$, $\epsilon > 0$, of algorithms for the n-chip problem, and we give powerful evidence that, on average in the biased case, they behave significantly better than the optimal algorithm for the majority problem. These algorithms are based on Chassaing's algorithm [6, Section 9].

The algorithm $A_n(\rho, \alpha, \epsilon)$ works by choosing a sample of size $m < n$ from the chips and combining the m chips into larger and larger happy binomial trees. Trees that are not happy are ignored because they cannot affect the majority of good chips. When a sufficiently large happy binomial tree is produced, its root must be a good chip or we would violate the condition that a strict majority of the chips is good. The difficulties here are the determination of the size of the sample and the analysis of the resulting process. The sample must be large enough that the likelihood of failure is exponentially small, but small enough to establish the necessary time bounds of the algorithm.

The algorithm processes the $m_0 = m$ chips in stages: First the m_0 chips ($\mathcal{B}_0$ trees) are combined pairwise into $\lfloor m_0/2 \rfloor$ binomial trees of two chips each ($\mathcal{B}_1$ trees) and the trees that are not happy are ignored, leaving $m_1 \leq \lfloor m_0/2 \rfloor$ happy $\mathcal{B}_1$ trees. The m_1 happy $\mathcal{B}_1$ trees are combined pairwise into $\lfloor m_1/2 \rfloor$ $\mathcal{B}_2$ trees of four chips each and the trees that are not happy are ignored, leaving $m_2 \leq \lfloor m_1/2 \rfloor$ happy $\mathcal{B}_2$ trees. This process continues until there is at most one happy binomial tree of each order. This algorithm is similar to that in Pelc and Upfal [11, Prop. 4.1], but their algorithm does twice as many tests because they always have pairs of chips test each other.

If there are any happy binomial trees left, let the largest one be of order K and have root R, otherwise, if each m_i is even, let $K = -\infty$. In the final stage of the algorithm, the roots of all the other happy binomial trees (if any) test R. At this point in the algorithm, at least half of the chips in ignored trees are bad and, if R were bad, then all chips in R's subtree would be bad, as are those in the other happy binomial trees whose roots diagnosed R as good in the final stage. If that amounts to more than $n/2$ bad chips, R must be a good chip. In other words, if R were bad, at least X_m chips would be bad, where

$$X_m = 2^K + \frac{1}{2}\left|\begin{matrix}\text{ignored}\\ \text{chips}\end{matrix}\right| + \left|\begin{matrix}\text{chips in happy binomial trees that}\\ \text{diagnose } R \text{ as good in the final stage}\end{matrix}\right|. \tag{4}$$

If there are no happy binomial trees left at the end of the stages of tests, or if the consequences of R being bad are not sufficient to reach a contradiction, we use the algorithm of [9] to identify a good chip. That algorithm uses n tests, but it will rarely need to be invoked.

The performance of the algorithm $A_n(\rho, \alpha, \epsilon)$ rests on the choice of the sample size m; the analysis depends on computing of the expected number of tests used and $\mathbf{Pr}(X_m > n/2)$, the probability that, if R were bad, the number of known bad chips at this point would be more than $n/2$, ending the algorithm.

With the proper choice of m, the number of tests used is small and the probability that the algorithm of [9] must be invoked is exponentially small.

When bad chips always lie ($\alpha = 1$), this algorithm solves the majority problem, provided that $X_m > n/2$, since R belongs to the majority. It differs from a majority problem's algorithm in only two ways: First, when we compare two components in the majority problem, the choice of the two elements to be compared, one from each component, does not matter, but in the chip problem both the testing chip and the tested chip must be the roots of their components. Second, the choice of m differs—it must be large enough to make $\mathbf{Pr}(X_m \le n/2)$ exponentially small, but not too large, since the average cost increases linearly in m. When $\alpha = 1$ the choice

$$m = \frac{n}{2p} + n^{1/2+\epsilon},$$

for some ϵ, $0 < \epsilon < 1/2$, results from basic considerations about the binomial distribution (see [6, Section 9]).

When $\alpha < 1$, the set of chips that would be bad if chip R were bad contains good chips and also some bad chips that behaved as good chips (they told the truth). Thus the apparent proportion of good chips is larger than the true proportion p and we can choose m to be smaller than $n/2p$. As a consequence, we will see that the average complexity *decreases* as α decreases from 1 to 0. This is surprising, because one would expect uncertainty in the diagnosis to cause trouble when $\alpha = 1/2$ (maximum uncertainty), but on the contrary, the average cost turns out to be smaller by $\Theta(n)$ for the chip problem than for the majority problem ($\alpha = 1$). The example of $n = 5$ for the unbiased case (Figures 4 and 5) illustrates this strange phenomenon. The choice of m and the resulting computation of the average performance of $A_n(\rho, \alpha, \epsilon)$ are thus much subtler and more intricate than for the majority problem.

We begin by defining p_i as the probability that a binomial tree $\mathcal{B}_i$ is happy and that the chip at its root is good, given that its two binomial subtrees $\mathcal{B}_{i-1}$ are happy. Similarly, define q_i as the probability that a binomial tree $\mathcal{B}_i$ is happy and that the chip at its root is bad, given that its two binomial subtrees $\mathcal{B}_{i-1}$ are happy. Set $\rho_i = q_i/p_i$. Clearly $p_0 = p$, $q_0 = q$, and $\rho_0 = \rho$. We have the conditional probabilities

$$\mathbf{Pr}(\text{chip } x \text{ is good} \,|\, x \text{ is at root of a happy } \mathcal{B}_i) = \frac{p_i}{p_i + q_i} = \frac{1}{1 + \rho_i},$$

and

$$\mathbf{Pr}(\text{chip } x \text{ is bad} \,|\, x \text{ is at root of a happy } \mathcal{B}_i) = \frac{q_i}{p_i + q_i} = \frac{\rho_i}{1 + \rho_i}.$$

Now let r_1 and r_2 be two chips, each at the root of a happy binomial tree $\mathcal{B}_i$, and suppose r_1 tests r_2. The probability that r_2 is good and becomes the root of a happy binomial tree $\mathcal{B}_{i+1}$ (that is, the probability that r_2 is good and that r_1 diagnoses it so) is

$$p_{i+1} = \frac{1}{(1 + \rho_i)^2} + \frac{(1 - \alpha)\rho_i}{(1 + \rho_i)^2},$$

where the first term is for both r_1 and r_2 good, and the second term is for r_1 bad but correctly diagnosing r_2. Similarly, the probability that r_2 is bad and becomes the root of a happy binomial tree $\mathcal{B}_{i+1}$ is

$$q_{i+1} = \frac{\alpha\rho_i^2}{(1+\rho_i)^2},$$

which is the probability that r_1 and r_2 are bad and that r_1 misdiagnoses r_2. Finally then,

$$\begin{aligned} \rho_{i+1} &= \frac{q_{i+1}}{p_{i+1}} \\ &= \frac{\alpha\rho_i^2}{1+(1-\alpha)\rho_i}. \end{aligned}$$

Therefore,

$$\rho_i = \begin{cases} \rho & i = 0, \\ \frac{\alpha\rho_{i-1}^2}{1+(1-\alpha)\rho_{i-1}} & i > 0. \end{cases} \tag{5}$$

Define τ_i as the probability that combining two happy trees of size 2^{i-1} results in a happy tree of size 2^i. Step $i+1$ of algorithm $A_n(\rho, \alpha, \epsilon)$ can thus be viewed as a sequence of $\lfloor m_i/2 \rfloor$ independent Bernoulli trials, each of them producing a happy tree $\mathcal{B}_{i+1}$ with probability

$$\begin{aligned} \tau_i &= p_{i+1} + q_{i+1} \\ &= \frac{1}{(1+\rho_i)^2} + \frac{(1-\alpha)\rho_i}{(1+\rho_i)^2} + \frac{\alpha\rho_i^2}{(1+\rho_i)^2} \\ &= 1 - \rho_i \frac{1+\alpha+\rho_i(1-\alpha)}{(1+\rho_i)^2}. \end{aligned} \tag{6}$$

It follows that,

PROPOSITION 1. *The conditional distribution of m_i, given that $m_{i-1} = k$, is the binomial distribution with parameters $\lfloor k/2 \rfloor$ and τ_{i-1}.* □

Now define

$$\sigma = \tau_0\tau_1\tau_2\cdots.$$

Induction using (5) gives

$$\rho_i \le (\alpha\rho)^{2^i}/\alpha,$$

for $i > 0$; (6) gives

$$1 - 2\rho_i \le \tau_i < 1, \tag{7}$$

and so

$$1 - \tau_i \le \frac{2}{\alpha}(\alpha\rho)^{2^i}. \tag{8}$$

Elementary arguments now give

$$0 < \sigma < 1. \tag{9}$$

Let

$$\lambda(\rho,\alpha)=\frac{1+\sigma}{2},$$

and

$$\mu(\rho,\alpha)=\frac{1}{2}+\frac{\tau_0}{4}+\frac{\tau_0\tau_1}{8}+\frac{\tau_0\tau_1\tau_2}{16}+\cdots.$$

We have $\lambda(\rho,1)=p$, $\mu(\rho,1)=2p\Phi(\rho)$ [see (1)], and $\lambda(\rho,\alpha)$ and $\mu(\rho,\alpha)$ are both decreasing in ρ and in α by elementary calculus:

Our sample size will be

$$m=\frac{n}{2\lambda(\rho,\alpha)-\epsilon}. \tag{10}$$

For ϵ small enough, this is a proper subset of the chips because, by (9), $\sigma>0$ and so $2\lambda(\rho,\alpha)>1$. We explain our choice of m as follows. As we study X_m, we will see that it behaves much like a binomially distributed random variable and that it does not deviate much from its mean, $\lambda(\rho,\alpha)m+o(m)$. Since we want it likely that $X_m>n/2$, we choose a sample slightly larger than $\frac{n}{2\lambda(\rho,\alpha)}$.

Finally, define

$$\begin{aligned}\Psi(\rho,\alpha) &= \frac{\mu(\rho,\alpha)}{2\lambda(\rho,\alpha)}\\ &= \frac{1}{1+\tau_0\tau_1\tau_2\cdots}\left(\frac{1}{2}+\frac{\tau_0}{4}+\frac{\tau_0\tau_1}{8}+\frac{\tau_0\tau_1\tau_2}{16}+\cdots\right);\end{aligned}$$

the function Ψ is plotted in Figure 6. In the appendix we will prove

THEOREM 1. *Let $c_n(\rho,\alpha,\epsilon)$ be the expected number of tests used by the algorithm $A_n(\rho,\alpha,\epsilon)$ under the probabilistic assumptions described at the beginning of this section. Then,*

$$\lim_{\epsilon\to 0}\lim_{n\to\infty}\frac{c_n(\rho,\alpha,\epsilon)}{n}=\Psi(\rho,\alpha).$$

COROLLARY 1.

$$\limsup_{n\to\infty}\frac{C_n(\rho,\alpha)}{n}\le\Psi(\rho,\alpha).$$

Note, from (1), that $\Psi(\rho,1)=\Phi(\rho)$; we conjecture that

CONJECTURE 1. *For $\alpha<1$ and $\rho>0$,*

$$\Psi(\rho,\alpha)<\Phi(\rho). \tag{11}$$

Figure 6 provides very powerful numerical evidence for this conjecture. A consequence of Conjecture 1 would be that

$$\limsup_{n\to\infty}\frac{C_n(\rho,\alpha)}{n}<\lim_{n\to\infty}\frac{C_n(\rho,1)}{n},$$

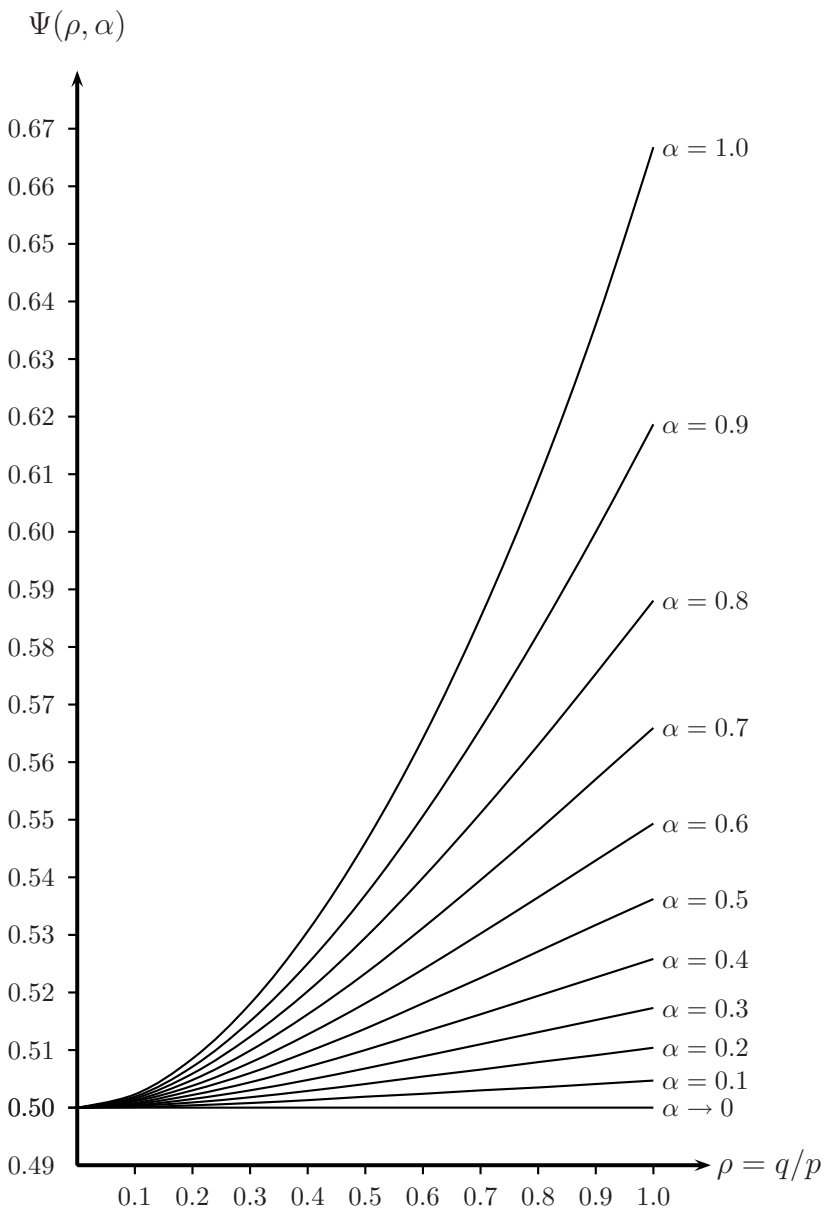

Figure 6: Plot of $\Psi(\rho, \alpha)$, the cost of $A_n(\rho, \alpha, \epsilon)$ per chip, for various values of α and ρ. We conjecture that for $\alpha < 1$ and $\rho > 0$, $\Psi(\rho, \alpha) < \Psi(\rho, 1)$.

for $\alpha < 1$.

Thus, the chip problem is not only different in the average case, but its average complexity is smaller by $\Theta(n)$. We conjecture further that

CONJECTURE 2.

$$\limsup_{n\to\infty} \frac{C_n(\rho,\alpha)}{n} = \Psi(\rho,\alpha).$$

Three observations support Conjecture 2. First, by our discussion of the choice of m as given in (10), it is clearly of minimal size. Second, no other algorithm uses tests as efficiently as $A_n(\rho,\alpha,\epsilon)$: given two happy $\mathcal{B}_{i-1}$ trees, a test of one root by the other has a high probability of giving a happy $\mathcal{B}_i$, contributing 2^i to X_m, while even if a non-happy $\mathcal{B}_i$ results, it still contributes 2^{i-1} to X_m at no additional cost. Third, if we choose all n chips as the sample, algorithm $A_n(\rho,1,\epsilon)$ is precisely the majority algorithm from [3] which is optimal in the worst case and within $O(\log n)$ of being optimal in the average case.

4.4 Discussion

We expect that when the probability α of an erroneous diagnosis increases from $\alpha = 0$, $\Psi(\rho,\alpha)$ increases with α. It would be surprising, however, if $\Psi(\rho,\alpha)$ were increasing in the neighborhood of $\alpha = 1$; rather, we expect the case $\alpha = 1$ (which is the majority problem) to be easier than $\alpha = 1-\varepsilon$ because uncertainty about the diagnosis (not knowing if it is a lie) should cause additional trouble, as we have seen: The proof that the chip problem is at least as complex as the majority problem ($\alpha = 1$) is obvious, but the proof of the converse relies on the special structure of the worst-case-optimal algorithm for the majority problem, an algorithm that is not average-case optimal. Despite this intuition, there is numerical evidence that $\Psi(\rho,\alpha)$ is increasing throughout the interval $[0,1]$ (see Figure 6), including the neighborhood of $\alpha = 1$.

The argument that the smaller the probability of a lie, the smaller the cost of the algorithm is simplistic. We have seen that the algorithm $A_n(\rho,\alpha,\epsilon)$ needs to work on a subset with size approximately $m = \frac{n}{2\lambda(\rho,\alpha)}$ in order to find a good chip, and that its cost is approximately $\mu(\rho,\alpha)m$, in which

$$\begin{aligned} \lambda(\rho,\alpha) &= \frac{1+\sigma}{2} \\ \mu(\rho,\alpha) &= \frac{1}{2} + \frac{\tau_0}{4} + \frac{\tau_0\tau_1}{8} + \frac{\tau_0\tau_1\tau_2}{16} + \cdots, \end{aligned}$$

so that

$$\Psi(\rho,\alpha) = \frac{\mu(\rho,\alpha)}{2\lambda(\rho,\alpha)}.$$

The decrease of $\mu(\rho,\alpha)$ makes the increase of $\Psi(\rho,\alpha)$ problematic. This surprising decrease of $\mu(\rho,\alpha)$ is due to an increasing number of bad chips misdiagnosing good chips, leading to an increasing number of unhappy trees, thus saving further tests. Nevertheless, $\Psi(\rho,\alpha)$ is actually increasing, and this can only be explained by a stronger decrease of $\lambda(\rho,\alpha)$.

This situation is reminiscent of the majority problem when ρ increases. The algorithm $A_n(\rho, 1, n^{-1/4})$, for instance, is known to be asymptotically average-case optimal for the majority problem [3], [6]. Here we can prove that $\mu(\rho, 1) = 2p\Phi(\rho)$ is increasing in ρ, meaning that the optimal algorithm works more slowly when the majority is more pronounced (that is, $\rho \in [0, \frac{1}{2}]$). This is surprising since our intuition tells us that a large majority makes the problem easier. On the other hand, the size ($\approx \frac{n}{\lambda(\rho,1)} = \frac{n}{2p}$) of the subset on which the algorithm works decreases when ρ increases. The average-case complexity of the majority problem is the product of a decreasing function (the speed—the ratio of the time needed to the size of the subset) by a increasing function (the size of the subset). It turns out that this product is increasing, so an oversimplified argument leads to the right conclusion, at the price of a superficial understanding of the problem.

The proof of monotonicity is easy for $\mu(\rho, 1)$ and $\lambda(\rho, 1)$, but more intricate for $\Phi(\rho) = \mu(\rho, 1)/\lambda(\rho, 1)$; it seems out of reach for $\mu(\rho, \alpha)$, $\lambda(\rho, \alpha)$ and $\Psi(\rho, \alpha)$, for which we have only numerical evidence.

4.5 An Alternative Model

Assume that the probability that a bad chip x lies depends on its working condition, say α_x. This yields the following model: for each bad chip x, choose the working condition α_x at random, the α_x being independently, identically distributed with a common probability distribution $\mu(du)$. We assume that a chip r_1 lies about the true nature of a chip r_2 with probability $f(\alpha_{r_1})$, and that r_1 tells the truth with probability $1 - f(\alpha_{r_1})$. As in the model described at the beginning of this subsection, the probability of a lie is thus a constant α, given by

$$\begin{aligned} \alpha &= \int \mathbf{Pr}(x \text{ lies about } y \,|\, \alpha_x = u)\mu(du) \\ &= \int f(u)\mu(du), \end{aligned}$$

but the status of two edges with the same origin x are no longer independent, since we have

$$\begin{aligned} \mathbf{Pr}(x \text{ lies about } y \text{ and } z) &= \int \mathbf{Pr}(x \text{ lies about } y \text{ and } z \,|\, \alpha_x = u)\mu(du) \\ &= \int f^2(u)\mu(du) \\ &> \alpha^2. \end{aligned}$$

Such dependence is natural because a positive correlation between different tests by the same chip is likely. Furthermore, the average case analysis of $A_n(\rho, \alpha, \epsilon)$ in this alternative model is identical to that of the preceding model, since $A_n(\rho, \alpha, \epsilon)$ forms binomial trees only and thus no chip ever tests more than one other chip.

5 Conclusions and Open Problems

We have shown that the chip problem and the modified majority problem are most likely different in the average case. We have designed an algorithm for the chip problem in the biased case whose average complexity appears sharply better than the average complexity of the optimal algorithm for the majority problem.

Many open problems remain, in particular proving Conjectures 1 and 2, finding average-case optimal algorithms for the chip problem, and clarifying the relationship between the worst case of the chip problem (the modified majority problem, that is) and that of the majority problem for even n.

Can algorithms be restricted so that only roots of components are involved in tests? It seems so since the information at any point of an algorithm is a bound on number good minus number bad for a component.

Appendix: Proof of Theorem 1

The analysis is easier in the independent model [11] than in our conditioned model, given $N > n/2$. The next proposition shows that the two models are so close that, for the level of accuracy that we need, we can use either one, justifying our assumption of independence of chip faultiness as in [11]:

PROPOSITION 2. *For any event A we have*

$$|\mathbf{Pr}[A] - \mathbf{Pr}[A \mid N > n/2]| \leq 2e^{-(p-q)^2 n/2}, \tag{12}$$

and similarly, for any bounded random variable X, $|X| \leq M$, we have

$$|\mathbf{E}[X] - \mathbf{E}[X \mid N > n/2]| \leq 2Me^{-(p-q)^2 n/2}. \tag{13}$$

Proof. We prove only (13); the proof of (12) is similar. Chernoff's inequality [5] states that, if N is a binomially distributed random variable with parameters n and p, then

$$\mathbf{Pr}(N \leq pn - h) \leq e^{-2h^2/n}. \tag{14}$$

Taking $h = (p-q)n/2$ in (14) we get

$$\mathbf{Pr}(N \leq n/2) \leq e^{-(p-q)^2 n/2}.$$

Now

$$\begin{aligned} \mathbf{E}[X] &= \mathbf{E}[X \mid N > n/2]\mathbf{Pr}(N > n/2) + \mathbf{E}[X \mid N \leq n/2]\mathbf{Pr}(N \leq n/2) \\ &= \mathbf{E}[X \mid N > n/2](1 - \mathbf{Pr}(N \leq n/2)) + \mathbf{E}[X \mid N \leq n/2]\mathbf{Pr}(N \leq n/2), \end{aligned}$$

so that

$$\mathbf{E}[X] - \mathbf{E}[X \mid N > n/2] = \mathbf{Pr}(N \leq n/2)\left(\mathbf{E}[X \mid N \leq n/2] - \mathbf{E}[X \mid N > n/2]\right).$$

Taking absolute values,

$$\begin{aligned} &|\mathbf{E}[X] - \mathbf{E}[X \mid N > n/2]| \\ &\quad \leq \quad \mathbf{Pr}(N \leq n/2)\,(|\mathbf{E}[X \mid N > n/2]| + |\mathbf{E}[X \mid N < n/2]|) \\ &\quad \leq \quad 2M\mathbf{Pr}(N \leq n/2), \end{aligned}$$

where the expectation $\mathbf{E}[X]$ is taken under the assumption of independent chip faultiness, and the expectation $\mathbf{E}[X \mid N > n/2]$ according to the distribution $p^k q^{n-k}/\mathbf{Pr}(N > n/2)$. □

The sample size $m = m_0$ must be large enough that the likelihood of failure, $\mathbf{Pr}(X_m < n/2)$, is exponentially small, but small enough to establish the necessary time bounds of the algorithm. To analyze the expression of X_m, recall from (4) that the largest happy binomial tree left in the final stage of the algorithm has root R and size 2^K. For $i \geq 0$, the $i+1$st step of algorithm $A_n(\rho, \alpha, \epsilon)$ forms m_i happy trees and hence $\lfloor m_i/2 \rfloor - m_{i+1}$ unhappy trees, each of size 2^i, thus and hence

$$X_m = 2^K + \frac{1}{2}\sum_{i \geq 0} 2^i(\lfloor m_i/2 \rfloor - m_{i+1}) + \sum_{\substack{i \geq 0,\ m_i \text{ odd} \\ B_i \text{ diagnoses } B_K \text{ as good}}} 2^i. \tag{15}$$

Similarly, the number T_m of tests performed by the algorithm is given by

$$T_m = \sum_{i \geq 0} \lfloor m_i/2 \rfloor + \sum_{i \geq 0,\ m_i \text{ odd}} 1. \tag{16}$$

From Proposition 1, the conditional expectation of $\lfloor m_{i-1}/2 \rfloor - m_i$, given m_i, is approximately

$$\frac{m_i(1 - \tau_{i+1})}{2}.$$

This last fact and relations (15) and (16) lead to the approximations in the following two propositions:

Proposition 3.

$$\mathbf{E}[X_m] = \lambda(\rho, \alpha)m + o(m), \tag{17}$$

and

$$\mathbf{E}[T_m] = \mu(\rho, \alpha)\ m + O(\log m). \tag{18}$$

The proof of Proposition 3 rests on three lemmas:

Lemma 1.

$$\begin{aligned} \mathbf{E}[m_i] &= s_i + O(1), \\ \mathbf{E}\left[\left\lfloor \frac{m_i}{2} \right\rfloor\right] &= \frac{1}{2}s_i + O(1), \end{aligned}$$

where

$$s_i = \begin{cases} m & i = 0, \\ s_{i-1}\tau_{i-1}/2 & i > 0; \end{cases}$$

that is, $s_i = \tau_0\tau_1\cdots\tau_{i-1}m/2^i$.

Proof. The conditional distribution of m_i, given m_{i-1}, is a binomial law with parameters $\lfloor\frac{m_{i-1}}{2}\rfloor$ and τ_i, for $i \geq 1$, thus

$$\mathbf{E}[m_i] = \tau_{i-1}\mathbf{E}\left[\left\lfloor\frac{m_{i-1}}{2}\right\rfloor\right]$$

and

$$\mathbf{E}[m_i] \leq \frac{\tau_{i-1}}{2}\mathbf{E}[m_{i-1}] \leq \tau_0\tau_1\cdots\tau_{i-1}m/2^i = s_i.$$

Also

$$\begin{aligned}\mathbf{E}[m_i] &\geq \frac{\tau_{i-1}}{2}\mathbf{E}[m_{i-1}-1]\\ &\geq \tau_0\tau_1\cdots\tau_{i-1}m/2^i - \left(\frac{\tau_{i-1}}{2}+\frac{\tau_{i-1}\tau_{i-2}}{4}+\cdots+\frac{\tau_{i-1}\tau_{i-2}\cdots\tau_0}{2^i}\right)\\ &\geq s_i - 1,\end{aligned}$$

giving the lemma. □

As a consequence of Proposition 1, Chernoff's inequality holds, so (14) tells us that for $i > 0$

$$\mathbf{Pr}(m_{i+1} \leq s\tau_i - h \,|\, \lfloor m_i/2\rfloor = s) \leq 2e^{-2h^2/s}.$$

Also, the conditional distribution of m_{i+1}, given that $m_i = 2s+t$, get stochastically larger as t increases, $t \geq 0$, and thus, for $i > 0$,

$$\mathbf{Pr}(m_{i+1} \leq s\tau_i - h \,|\, m_i = 2s+t) \leq 2e^{-2h^2/s}.$$

Therefore for $i > 0$,

$$\begin{aligned}&\mathbf{Pr}(m_{i+1} \leq s\tau_i - h \text{ and } m_i \geq 2s)\\ &\quad= \sum_{t=0}^{\infty}\mathbf{Pr}(m_{i+1} \leq s\tau_i - h \,|\, m_i = 2s+t)\mathbf{Pr}(m_i = 2s+t)\\ &\quad\leq 2e^{-2h^2/s}.\end{aligned} \tag{19}$$

Lemma 2. *Let*

$$h_i = \begin{cases} m^{2/3} & i = 0,\\ h_{i-1}\tau_{i-1} & i > 0,\end{cases}$$

and s_i *be as defined in Lemma 1. Then*

$$\mathbf{Pr}(m_i < s_i - h_i) \leq 2ie^{-\sigma m^{1/3}/8}, \tag{20}$$

so that m_i *does not vary much from its expected value.*

Proof. By (19), for $i > 0$ we have

$$\begin{aligned}&\mathbf{Pr}(m_i < s_i - h_i)\\ &\quad\leq \mathbf{Pr}(m_i < s_i - h_i \text{ and } m_{i-1} \geq s_{i-1} - h_{i-1}) + \mathbf{Pr}(m_{i-1} < s_{i-1} - h_{i-1})\\ &\quad\leq 2\exp\left(\frac{-h_i^2}{4(s_{i-1}-h_{i-1})}\right) + \mathbf{Pr}(m_{i-1} < s_{i-1} - h_{i-1}),\\ &\quad\leq 2e^{-\sigma m^{1/3}/8} + \mathbf{Pr}(m_{i-1} < s_{i-1} - h_{i-1}),\end{aligned} \tag{21}$$

because

$$\frac{h_i^2}{4(s_{i-1}-h_{i-1})} \geq \frac{h_i^2}{4s_{i-1}} = m^{1/3}2^{i-3}(\tau_0\tau_1\tau_2\cdots\tau_{i-1})\tau_{i-1} \geq \frac{m^{1/3}\sigma}{8},$$

since $i > 0$ and

$$1 \geq \tau_i \geq \frac{1}{1+\rho_i} \geq 1/2.$$

With the base case

$$\mathbf{Pr}(m_0 < s_0 - h_0) = \mathbf{Pr}(m < m - m^{2/3}) = 0,$$

a simple induction on (21) completes the proof. □

Let $S_0 = \{r_1, r_2, \ldots, r_{m_i}\}$ denote the set of roots of happy trees of size 2^i present at the beginning of stage i.

LEMMA 3. *The probability that all tests involving two roots of S_0 are happy, is bounded below by*

$$1 - \frac{2}{\alpha}(\alpha\rho)^{2^i} m/2^i,$$

provided that $\frac{1}{\alpha}(\alpha\rho)^{2^i} \leq 1$ (which holds for i large enough).

Proof. If the tests performed at stages i, $i+1$, ..., and the tests of the final stage are all happy, it gives exactly $m_i - 1$ tests involving two elements of S_0, but there are eventually fewer than $m_i - 1$ such tests if all the diagnosis are not happy. We prove that the conditional probability $p_{i,k}$ that there are $m_i - 1$ such happy diagnosis, given $m_i - 1 = k$, satisfies

$$1 - p_{i,k} \leq \frac{2k}{\alpha}(\alpha\rho)^{2^i}, \tag{22}$$

giving the proposition, since $m_i \leq m/2^i$. Let π_ℓ denote the conditional probability that the ℓth test among these k tests is happy, given that the preceding $\ell - 1$ tests are happy too. We have $p_{i,k} = \prod_{\ell=1}^{k} \pi_\ell$, so relation (22) follows from

$$1 - \pi_\ell \leq 2\beta_i = \frac{2}{\alpha}(\alpha\rho)^{2^i}, \tag{23}$$

which we now prove.

Let S_ℓ be the set of elements of S_0 that are still at the root of a happy (eventually non binomial) tree after the ℓth test. For $x \in S_\ell$, let $p(x,\ell)$ denote the conditional probability that x is good, given the result of the ℓ first tests; let $q(x,\ell)$ denote the corresponding conditional probability that x is bad, given the result of the ℓ first tests. Define the rate $\rho_{x,\ell} = q(x,\ell)/p(x,\ell)$. By induction on ℓ we have that, for all $x \in S_\ell$,

$$\rho_{x,\ell} \leq \beta_i.$$

The induction property holds true for $\ell = 0$. Assume that the induction property holds true for a general ℓ, and that in the $(\ell+1)$st test, chip y tests chip z. As a consequence of the induction property for ℓ, relation $\rho_{x,\ell} \le \beta_i$ holds for $x \in S_{\ell+1} - \{z\}$. The conditional probability that the $(\ell+1)$st test is happy and that z is good is

$$\frac{1+(1-\alpha)\rho_{y,\ell}}{(1+\rho_{y,\ell})(1+\rho_{z,\ell})};$$

the conditional probability that the $(\ell+1)$st test is happy and that z is bad is

$$\frac{\alpha\rho_{y,\ell}\rho_{z,\ell}}{(1+\rho_{y,\ell})(1+\rho_{z,\ell})}.$$

Thus

$$\begin{aligned}\rho_{z,1+\ell} &= \frac{\alpha\rho_{y,\ell}\rho_{z,\ell}}{1+(1-\alpha)\rho_{y,\ell}} \\ &\le \beta_{i+1} \le \beta_i,\end{aligned}$$

giving the induction. Finally, the conditional probability that the $(\ell+1)$st test is unhappy is given by

$$\begin{aligned}1-\pi_\ell &= 1 - \frac{1+(1-\alpha)\rho_{y,\ell}+\alpha\rho_{y,\ell}\rho_{z,\ell}}{(1+\rho_{y,\ell})(1+\rho_{z,\ell})} \\ &= \frac{\rho_{z,\ell}+\alpha\rho_{y,\ell}+(1-\alpha)\rho_{y,\ell}\rho_{z,\ell}}{(1+\rho_{y,\ell})(1+\rho_{z,\ell})} \\ &\le \rho_{z,\ell}+\rho_{y,\ell} \le 2\beta_i.\end{aligned}$$

□

Proof of Proposition 3. By Lemma 3, the probability that $K = -\infty$, and that the $\Theta(n)$ algorithm of [9] is needed, is exponentially small, so we ignore it in our estimates of T_m. Note that $K \le \lg m$, and that $m_i = 0$ if $i > K$. Relation (16) leads to

$$\sum_{i=0}^{K} \left\lfloor \frac{m_i}{2} \right\rfloor \le T_m \le \frac{1}{2} \sum_{i=0}^{\lceil \lg m \rceil} m_i + \lfloor \lg m \rfloor, \tag{24}$$

because $\lfloor m_i/2 \rfloor$ tests are made at every stage except the final stage which makes at most $\lfloor \lg m \rfloor$ tests. By (24) and Lemma 1,

$$\mathbf{E}[T_m] = \frac{1}{2} \sum_{i=0}^{\lceil \lg m \rceil} s_i + O(\log m),$$

and therefore

$$\mathbf{E}[T_m] = m \sum_{i=0}^{\lceil \lg m \rceil} \tau_0\tau_1\cdots\tau_{i-1}/2^{i+1} + O(\log m),$$

proving (18) as claimed. The proof of (17) is similar, using (15). □

PROPOSITION 4. *For any positive numbers ϵ and θ ($\theta < 1/3$),*

$$\mathbf{Pr}\big(X_m < (\lambda(\rho,\alpha) - \epsilon)m\big) = o\Big(\exp(-m^\theta)\Big);$$

that is, X_m does not deviate much from its mean.

Proof. We apply Lemma 1 to a carefully chosen value of i. Inequality (20) holds for any $i \geq 1$, but it is of use to us only when $h_i = o(s_i)$, that is, when $m^{2/3} = o(m2^{-i})$, or equivalently, $2^i = o(m^{1/3})$. It suffices to take $i \leq \delta \lg m$ for $0 < \delta < 1/3$. For such i, (20) becomes

$$\mathbf{Pr}(m_i < s_i - h_i) \leq (2\delta \lg m) e^{-\sigma m^{1/3}/8}. \tag{25}$$

By Lemma 3 the probability that, during and after stage $i = \lfloor \delta \lg m \rfloor$, the tests involving roots of happy trees with size at least 2^i all give the diagnosis "good", is at least

$$1 - \frac{\frac{2}{\alpha}(\alpha\rho)^{2^{\lfloor \delta \lg m \rfloor}} m}{2^{\lfloor \delta \lg m \rfloor}} \geq 1 - \frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta}.$$

In that case,

$$K = i + \lfloor \lg m_i \rfloor,$$

and at the end of the algorithm, the tree with root R contains all the chips that were in trees of order i at stage i, at least $2^i m_i$ chips. If R were bad, there would thus be at least $2^i m_i$ bad chips, and at most $2^i - 1$ good chips (the aggregate of all the chips in small trees *not* in R that diagnosed R as bad in the final stage—they are in happy binomial trees with fewer than 2^i chips, at most one of each order 0 to $i-1$), with a probability at least

$$1 - \frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta}.$$

That is, if R were bad, of the sample of m chips, we would know that $2^i m_i$ were bad, $2^i - 1$ were good or bad, and the remaining $m - (2^i m_i + 2^i - 1)$ ignored chips were at least half bad, so the number of bad chips would be at least

$$2^i m_i + \frac{m - (2^i m_i + 2^i - 1)}{2} = \frac{m + 2^i m_i - 2^i + 1}{2}.$$

Hence

$$\mathbf{Pr}\left(X_m \geq \frac{m + 2^i m_i - 2^i + 1}{2}\right) \geq 1 - \frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta},$$

and so,

$$\mathbf{Pr}\left(X_m < \frac{m + 2^i m_i - 2^i + 1}{2}\right) < \frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta}. \tag{26}$$

Using the identity

$$\begin{aligned} \mathbf{Pr}(u < v) &= \mathbf{Pr}((u < v \text{ and } v \leq w) \text{ or } (u < v \text{ and } v > w)) \\ &\leq \mathbf{Pr}(u < v \text{ and } v \leq w) + \mathbf{Pr}(u < v \text{ and } v > w) \\ &\leq \mathbf{Pr}(u \leq w) + \mathbf{Pr}(w < v) \end{aligned}$$

with $u = X_m$, $v = \frac{m+2^i(s_i-h_i)-2^i+1}{2}$, and $w = \frac{m+2^i m_i-2^i+1}{2}$ gives us

$$\begin{aligned}
&\mathbf{Pr}\left(X_m < \frac{m+2^i(s_i-h_i)-2^i+1}{2}\right)\\
&\quad\le\ \mathbf{Pr}\left(X_m \le \frac{m+2^i(s_i-h_i)-2^i+1}{2}\right)\\
&\qquad +\mathbf{Pr}\left(\frac{m+2^i m_i-2^i+1}{2} < \frac{m+2^i(s_i-h_i)-2^i+1}{2}\right);
\end{aligned}$$

but the latter probability simplifies to yield

$$\begin{aligned}
&\mathbf{Pr}\left(X_m < \frac{m+2^i(s_i-h_i)-2^i+1}{2}\right)\\
&\quad\le\ \mathbf{Pr}\left(X_m \le \frac{m+2^i(s_i-h_i)-2^i+1}{2}\right) + \mathbf{Pr}(s_i-h_i < m_i)\\
&\quad\le\ \frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta} + (2\delta\lg m)e^{-\sigma m^{1/3}/8}
\end{aligned}$$

by (26) and (25). Taking the complement gives us

$$\begin{aligned}
&\mathbf{Pr}\left(X_m \ge \frac{m+2^i(s_i-h_i)-2^i+1}{2}\right)\\
&\qquad\ge\ 1-\frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta} - (2\delta\lg m)e^{-\sigma m^{1/3}/8}. \qquad (27)
\end{aligned}$$

Now

$$\begin{aligned}
&\frac{m+2^i(s_i-h_i)-2^i+1}{2}\\
&\quad=\ m\left(\lambda(\rho,\alpha)+\frac{\tau_0\tau_1\cdots\tau_{i-1}(1-\tau_i\tau_{i+1}\cdots)}{2}\right)-\frac{2^i h_i+2^i-1}{2}. \qquad (28)
\end{aligned}$$

Elementary calculation yields

$$\frac{\tau_0\tau_1\cdots\tau_{i-1}(1-\tau_i\tau_{i+1}\cdots)}{2} \le \tau_0[\rho_i+\rho_{i+1}+\cdots] = o(1),$$

and

$$2^i h_i + 2^i = O(2^{\delta\lg m}\tau_0\tau_1\cdots\tau_{i-1}m^{2/3}) = o(m)$$

by our choice of δ. Thus (28) becomes

$$\begin{aligned}
\frac{m+2^i(s_i-h_i)-2^i+1}{2} &\ge \lambda(\rho,\alpha)m - o(m)\\
&> (\lambda(\rho,\alpha)-\epsilon)m,
\end{aligned}$$

for all $\epsilon > 0$ as m (that is, n) gets large. Now chosing δ in (27) so that $\theta < \delta < 1/3$ yields Proposition 4. □

□

The analysis of $A_n(\rho, \alpha, \epsilon)$ rests on Propositions 3, 4.

Since we want it likely that $X_m > n/2$, we choose a sample slightly larger than $\frac{n}{2\lambda(\rho,\alpha)}$. For instance the sample size

$$m_0 = \frac{n}{2\lambda(\rho, \alpha) - \epsilon}$$

does the job: with this sample size X_m is larger than $n/2$, and chip R is good, except with an exponentially small (in a power of n) probability. The average number of tests required by the algorithm $A_n(\rho, \alpha, \epsilon)$ thus satisfies

$$\lim_{n\to\infty} \frac{\mathbf{E}[T_m]}{n} = \frac{\mu(\rho, \alpha)}{2\lambda(\rho, \alpha) - \epsilon},$$

for any $\epsilon > 0$, and Theorem 1 follows.

References

[1] L. Alonso, P. Chassaing, E. M. Reingold, and R. Schott, The Worst-Case Chip Problem, *Info. Proc. Let.* **89** (2004), 303–308.

[2] L. Alonso, E. M. Reingold, and R. Schott, Determining the majority, *Info. Proc. Let.* **47** (1993), 253–255.

[3] L. Alonso, E. M. Reingold, and R. Schott, The average-case complexity of determining the majority, *SIAM J. Comput.* **26** (1997), 1–14.

[4] P. M. Blecher, On a logical problem, *Discrete Math.* **43** (1983), 107–110.

[5] B. Bollobas, *Random Graphs*, Academic Press, London, 1985.

[6] P. Chassaing, Determining the majority: the biased case, *Ann. Appl. Prob.* **7** (1997), 523–544.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[8] D. H. Greene and D.E. Knuth, *Mathematics for the Analysis of Algorithms*, 3rd ed., Birkhäuser, Boston, 1990.

[9] S. L. Hakimi and E. F. Schmeichel, An adaptive algorithm for system level diagnosis, *J. Algorithms* **5** (1984) 526–530.

[10] S. Lee and K. G. Shin, Probabilistic diagnosis of multiprocessor systems, *ACM Comp. Surv.* **26** (1994), 121–139.

[11] A. Pelc and E. Upfal, Reliable fault diagnosis with few tests, *Combin. Probab. Comput.* **7** (1998), no. 3, 323–333.

[12] F. Preparata, G. Metze, and R. T. Chien, On the connection assignment problem for diagnosable systems, *IEEE Trans. Comput.* **16** (1967) 848–854.

[13] M. E. Saks and M. Werman, On computing majority by comparisons, *Combinatorica* **11** (1991), 383–387.

[14] E. F. Schmeichel, S. L. Hakimi, M. Otsuka, and G. Sullivan, A parallel fault identification algorithm, *J. Algorithms* **11** (1990), 231–241.

[15] R. Smullyan, *What Is the Name of This Book?—The Riddle of Dracula and Other Logical Puzzles*, Prentice Hall, Englewood Cliffs, New Jersey, 1978.

[16] J. Vuillemin, A data structure for manipulating priority queues, *Comm. ACM* **21** (1978), 309–315.

[17] G. Weiner, Search for a majority element, *J. Statistical Planning and Inference* **100** (2002), 313–318.