The Chip Problem

Laurent Alonso, Philippe Chassaing, Edward M. Reingold, René Schott

The Chip Problem*

Laurent Alonso† Philippe Chassaing‡ Edward M. Reingold§
René Schott¶

September 3, 1998

Abstract. The "chip problem" is a fault diagnosis problem in which 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 $1-\alpha$. We show that the chip problem is closely related to a modified majority problem in the *worst case* and use this fact to obtain upper and lower bounds on algorithms for the chip problem. 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 (in which p is the probability that a chip is bad) whose average performance is better than the average cost of the best algorithm for the biased majority problem.

Key words. Fault diagnosis, algorithm analysis, chip problem, majority problem, probabilistic algorithms.

AMS(MOS) subject classifications. 68Q25, 68P10, 68Q20, 68R05, 05A10, 11A63

---

*This research was supported in part by INRIA and the NSF, through grant numbers NSF INT 90-16958 and 95-07248.

†INRIA-Lorraine and LORIA, Université Henri Poincaré-Nancy I, BP 239, 54506 Vandoeuvre-lès-Nancy, France. Email: `Laurent.Alonso@loria.fr`

‡Institut Elie Cartan, Université Henri Poincaré-Nancy I, BP 239, 54506 Vandoeuvre-lès-Nancy, France. Email: `Philippe.Chassaing@antares.iecn.u-nancy.fr`

§Department of Computer Science, University of Illinois at Urbana-Champaign, 1304 W. Springfield Avenue, Urbana, Illinois 61801, USA. 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, Rehovot, Israel. Email: `reingold@cs.uiuc.edu`

¶Institut Elie Cartan and LORIA, Université Henri Poincaré-Nancy I, BP 239, 54506 Vandoeuvre-lès-Nancy, France. Email: `Rene.Schott@loria.fr`

1 Introduction

In system diagnosis, according to [10],

> ...in a set U of n units (processors, modules, etc.) at most t are faulty, and 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. $u_i \in U$ might be asked to determine if u_j 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 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 refer to "chips" rather than "units" or "components" [5, exercise 4-7, page 75]. An algorithm can determine whether a bad (faulty) chip exists if and only if a strict majority of the chips are fault-free [8]. 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. In this paper we address only the 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. Other algorithmic questions are discussed in the conclusions.

2 Majority Problems

The *majority problem* of [1], [2], and [9] 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

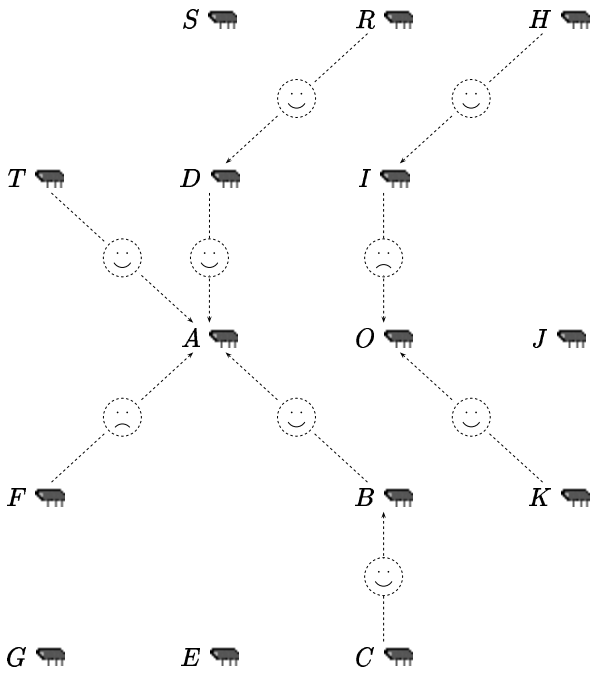

Figure 1: Sample test configuration of fifteen chips. We show a link from one chip to another labeled by a smiling face if the chip at the base of the arrow diagnoses the chip at the point of the arrow as good, and label the link with a frowning face if the diagnosis is bad. Chip F must be bad and chip 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.

or red, by pairwise equal/not equal color comparisons. When n is even, we must 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 [6], $\nu(n)$ is the number of 1-bits in the binary representation of n. This result was first proved by Saks and Werman [9]; [1] gave a short, elementary proof. [2] 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, [2] 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 [4]. 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

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

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

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 The Chip Problem in the Worst Case

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* [12] (see also [5, 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.

3.1 Faulty Chips Always Lie

Suppose faulty chips *always* give wrong answers. This problem is reminiscent of the Smullyan's puzzles [11] in which one lands on an island of truth-tellers and liars and must get correct information by asking questions to people of unknown type.

The problem of determining a good chip under such conditions is equivalent to the modified majority problem described above. The equivalence is based on

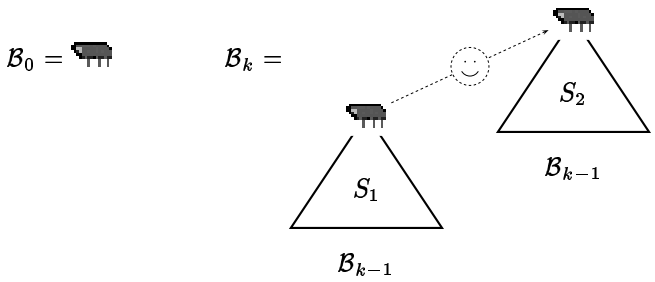

Figure 2: Binomial trees of chips. The smiling face indicates the chip at the base of the arrow says that the chip at the point of the arrow is good. If all arrows in the trees S_1 and S_2 are smiling and the chip at the root of S_2 is bad, all chips in both S_1 and S_2 must be bad by induction. Since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

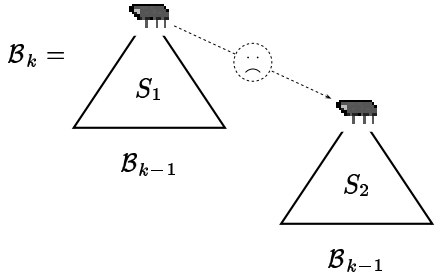

Figure 3: The frowning face indicates the chip at the base of the arrow says that the chip at the point of the arrow is bad. Suppose all the arrows in the trees S_1 and S_2 are smiling. If the chip at the root of S_1 is good, all chips in S_2 must be bad; if the chip at the root of S_1 is bad, all chips 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.

the following observation: When some chip x tells you that some other chip y is bad, you can conclude that y is bad, if you know that x is good; you can conclude that y is good, if you know that x is bad. Thus you can conclude, in any case, that x and y belong to different categories. Similarly, when some chip x tells you that some other chip y is good, you can conclude that x and y belong to the same category. But this is just a modification of the majority problem: Thus an algorithm for this *modified majority problem* serves equally well for the chip problem when bad chips always give wrong answers. Similarly, in the opposite direction, an algorithm for this chip problem serves equally well for the modified majority problem; this follows from the more general result we prove in the next section. Thus, when faulty chips always lie the chip problem is identical to the modified majority problem.

3.2 Faulty Chips Sometimes Lie

When faulty chips sometimes lie, the problem of determining a good chip is more intricate. Because any algorithm that solves the chip problem when faulty chips *sometimes* give wrong answers, solves the problem when faulty chips *always* lie, the chip problem when faulty chips *sometimes* lie is at least as complex in the worst case as the modified majority problem.

In fact, these two problems are *equivalent* in the worst case. Suppose there is an algorithm that determines the majority color in the modified majority problem, the same algorithm can be used to solve the chip problem. To prove this, we recall from [2] that any algorithm for the (modified) majority problem corresponds to a decision tree in which the state of affairs at any node in decision tree for an algorithm solving the majority problem can be described with a non-increasing sequence of positive integers

$$\Delta_1 \geq \Delta_2 \geq \cdots \geq \Delta_m > 0,$$

each Δ_i is the excess of one color over the other in subsets A_i, B_i where $A_1 \cup B_1 \cup A_2 \cup B_2 \cup \cdots \cup A_m \cup B_m$ is a partition of the n elements and the color of A_i is known to differ from that of B_i.

The state of the modified majority algorithm at the root of the tree is thus described by a sequence of n ones, corresponding to singleton sets A_i and empty sets B_i. At an internal node $(\Delta_1, \Delta_2, \cdots, \Delta_m)$, a color comparison $x_u : x_v$, $x_u \in A_i \cup B_i$ and $x_v \in A_j \cup B_j$, results in the appropriate combining of A_i, B_i, A_j, and B_j, the deletion of the two values Δ_i and Δ_j from the sequence, and the insertion of $\Delta_i + \Delta_j$ or $|\Delta_i - \Delta_j| > 0$ into place in the sequence, one in the left subtree the other in the right subtree, respectively, depending on the result of the color comparison—in this case the sequence shrinks in length by 1 in passing to subtrees. However, zeroes are never in the sequence (since $\Delta_i = 0$ would mean $|A_i| = |B_i|$), so if $\Delta_i - \Delta_j = 0$, the sequence for that subtree shrinks in length by 2. In other words, each internal node of the decision tree can have either both of its children with a Δ-vector one shorter in length than its own Δ-vector or its left child with a Δ-vector one shorter in length than its own and its right child with a Δ-vector two shorter in length than its own.

A leaf in the decision tree corresponds to an outcome of the algorithm, and hence the associated Δ-vector contains enough information to determine the majority. Since there must be a strict majority, we have

$$\Delta_1 > \sum_{i \geq 2} \Delta_i, \tag{1}$$

in which case any element from the set A_1 is a member of the majority.

To show that any algorithm for the modified majority problem corresponds to an algorithm using the same worst-case number of queries for the chip problem when chips sometimes lie, we give an interpretation of the decision tree for the algorithm that uses the same decision tree to solve the chip problem: At any point in the algorithm, a subset of the n chips is partitioned into sets $S_1, S_2, \ldots, S_m$ with each set S_i having *root* element $r_i \in S_i$. If chip r_i is good, the number of good chips in S_i minus the number of bad chips in S_i is *at most* Δ_i. On the other hand, if chip r_i is bad, the number of bad chips in S_i minus the number of good chips in S_i is *at least* Δ_i. Letting $G(S)$ and $B(S)$ be the number of good and bad chips in S, respectively, we have

$$|G(S_i)| - |B(S_i)| \leq \Delta_i \quad \text{if } r_i \text{ is good,} \tag{2}$$

$$|B(S_i)| - |G(S_i)| \geq \Delta_i \quad \text{if } r_i \text{ is bad,} \tag{3}$$

for $i = 1, 2, \ldots, m$. We can rewrite condition (3) as

$$|G(S_i)| - |B(S_i)| \leq -\Delta_i \quad \text{if } r_i \text{ is bad.} \tag{4}$$

Inequalities (2) and (4) tell us that, at any node in the tree, the number of good chips minus the number of bad chips overall is at most

$$\pm\Delta_1 \pm \Delta_2 \pm \Delta_3 \pm \cdots \pm \Delta_m,$$

with a plus sign for Δ_i if the root of S_i is good and minus sign if it is bad.

Initially, the Δ-vector consists of n ones representing n singleton sets of chips, each of which is the root of its set. If that one chip x (the root) is good, clearly $|G(\{x\})| - |B(\{x\})| = 1 - 0 = 1 \leq 1$ as needed. Similarly, if that one chip x (the root) is bad, clearly $|B(\{x\})| - |G(\{x\})| = 1 - 0 = 1 \geq 1$ as needed.

Suppose that at an internal node in the decision tree the modified majority algorithm makes for a color comparison $x_u : x_v$, $x_u \in A_i \cup B_i$ and $x_v \in A_j \cup B_j$. Let the node have Δ-vector $(\Delta_1, \Delta_2, \cdots, \Delta_m)$. We have the root r_i of S_i test r_j of S_j. The left child of the node, followed if r_i says r_j is good, has the same partition as the node, except S_i and S_j are combined into $S_i \cup S_j$ with root r_j and Δ_i and Δ_j are combined into $\Delta_i + \Delta_j$. The right child of the node, followed if r_i says r_j is bad, also has S_i and S_j combined into $S_i \cup S_j$ but with root r_i and Δ_i and Δ_j are combined into $\Delta_i - \Delta_j$, unless $\Delta_i - \Delta_j = 0$ in which case both S_i and S_j are discarded, as are Δ_i and Δ_j. The intuition in this latter case is that between them, S_i and S_j have more bad chips than good chips, so discarding them increases the majority of good chips in the sets remaining. Figure 4 depicts the interpretation just described.

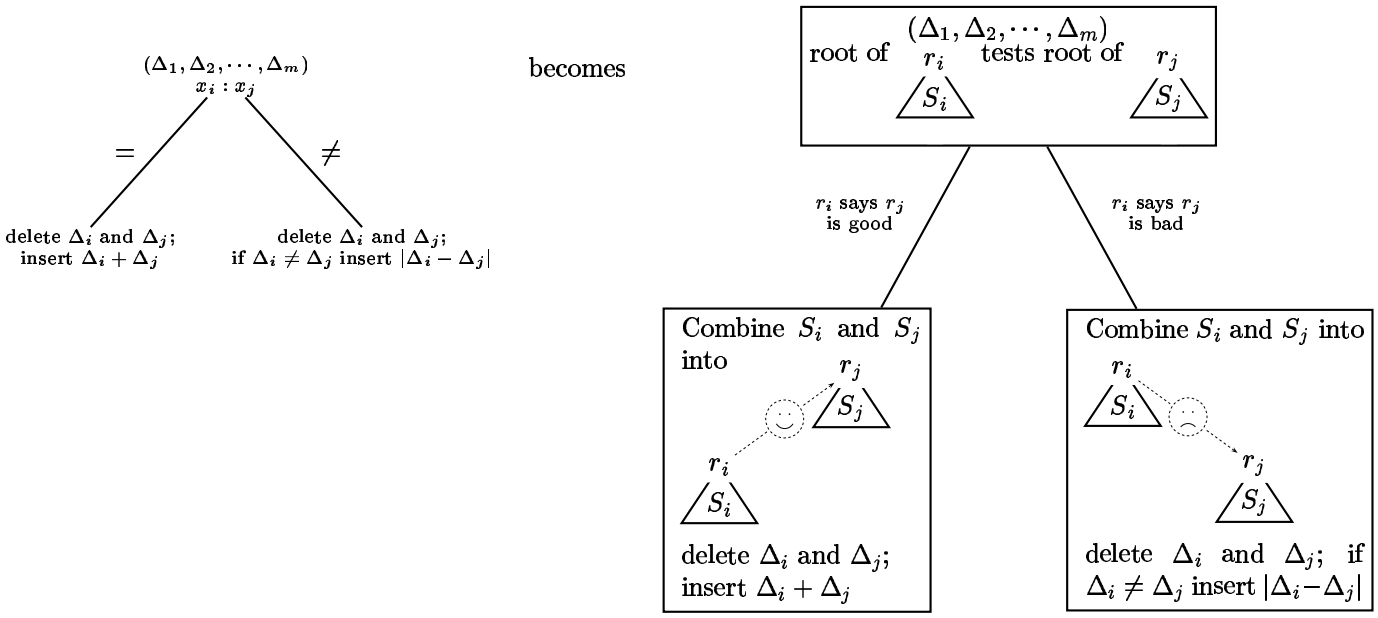

Figure 4: Interpretation of a decision tree for the modified majority problem as an algorithm for the chip problem.

First, consider the left child of the node (see Figure 4); that is, suppose r_i reports that r_j is good. If r_j is bad then r_i must be bad; since both r_i and r_j are bad, condition (3) tells us that

$$|B(S_i)| - |G(S_i)| \geq \Delta_i$$

and

$$|B(S_j)| - |G(S_j)| \geq \Delta_j$$

so that

$$|B(S_i \cup S_j)| - |G(S_i \cup S_j)| \geq \Delta_i + \Delta_j,$$

as needed since the root r_j of $S_i \cup S_j$ is bad. If r_j is good, condition (2) tells us that

$$|G(S_j)| - |B(S_j)| \leq \Delta_j,$$

and, then r_i can be either good or bad; if r_i is bad, condition (3), multiplied by -1, tells us that

$$|B(S_i)| - |G(S_i)| \leq -\Delta_i \leq \Delta_i;$$

if r_i is good, condition (2) tells us

$$|G(S_i)| - |B(S_i)| \leq \Delta_i.$$

In either case, then

$$|G(S_i \cup S_j)| - |B(S_i \cup S_j)| \leq \Delta_i + \Delta_j,$$

as needed since the root r_j of $S_i \cup S_j$ is good.

Now, consider the right child of the node (again, see Figure 4); that is, suppose r_i reports that r_j is bad. If r_i is good then r_j must be bad; since r_i is good, condition (2) tells us that

$$|G(S_i)| - |B(S_i)| \leq \Delta_i,$$

and since r_j is bad, condition (3), multiplied by -1, tells us that

$$|G(S_j)| - |B(S_j)| \leq -\Delta_j,$$

so that

$$|G(S_i \cup S_j)| - |B(S_i \cup S_j)| \leq \Delta_i - \Delta_j,$$

as needed since the root r_j of $S_i \cup S_j$ is good. If r_i is bad then condition (3) tells us that

$$|B(S_i)| - |G(S_i)| \geq \Delta_i.$$

r_j can be either good or bad; if it is good, condition (2), multiplied by -1, tells us that

$$|B(S_j)| - |G(S_j)| \geq -\Delta_j;$$

if r_j is bad, condition (3), tells us that

$$|B(S_j)| - |G(S_j)| \geq \Delta_j \geq -\Delta_j,$$

so that in either case

$$|B(S_i \cup S_j)| - |G(S_i \cup S_j)| \geq \Delta_i - \Delta_j,$$

as needed since the root r_j of $S_i \cup S_j$ is bad.

At a leaf in the decision tree for the modified majority problem we have the end condition

$$\Delta_1 > \sum_{i \geq 2} \Delta_i,$$

which can be written as

$$-\Delta_1 + \Delta_2 + \Delta_3 + \cdots + \Delta_m < 0.$$

If the root of S_1 is bad, the number of bad chips minus the number of good chips overall is at most

$$-\Delta_1 + \Delta_2 + \Delta_3 + \cdots + \Delta_m < 0,$$

which is impossible. Hence the root of S_1 must be a good chip.

4 The Chip Problem in the Average Case

The previous sections 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.

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 $1 - \alpha$. The problem is trivial when $\alpha = 0$ (that is, bad chips never lie) and we have already discussed the case $\alpha = 1$ (bad chips always lie), showing it is equivalent to the modified majority problem.

4.1 The Unbiased Case

In the *unbiased case*, we assume 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 5 (which is derived from the algorithm described in the conclusions of [2]), is optimal for the modified majority problem with $n = 5$, having cost 2.25. On the other hand, Figure 6 shows a similar 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 $0 < \alpha < 1$.

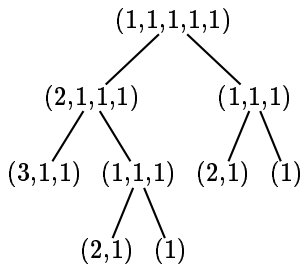

Figure 5: Optimal tree for $n = 5$ in the majority problem [2]. It has expected cost 2.25.

4.2 The Biased Case

In the *biased case*, each chip is good independently with known probability p and bad with probability $q = 1 - p < p$, and suppose $N > n/2$ of the n chips are good. We assume that the distribution of good chips is the conditional distribution given that $N > n/2$; that is, we assume that each partition of the chips into k good chips and $n - k$ bad chips, $k > n/2$, has probability $p^k q^{n-k}/\mathbf{Pr}(N > n/2)$, where

$$\mathbf{Pr}(N > n/2) = \sum_{n/2 < k \leq n} \binom{n}{k} p^k q^{n-k}$$

Since $q < p$, $\rho = q/p < 1$.

Let $C_n(\rho, \alpha)$ be the average-case complexity for the n-chip problem; that is, $C_n(\rho, \alpha)$ is the 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 [4, 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 of since 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 $\lfloor m_1/2 \rfloor$ $\mathcal{B}_2$

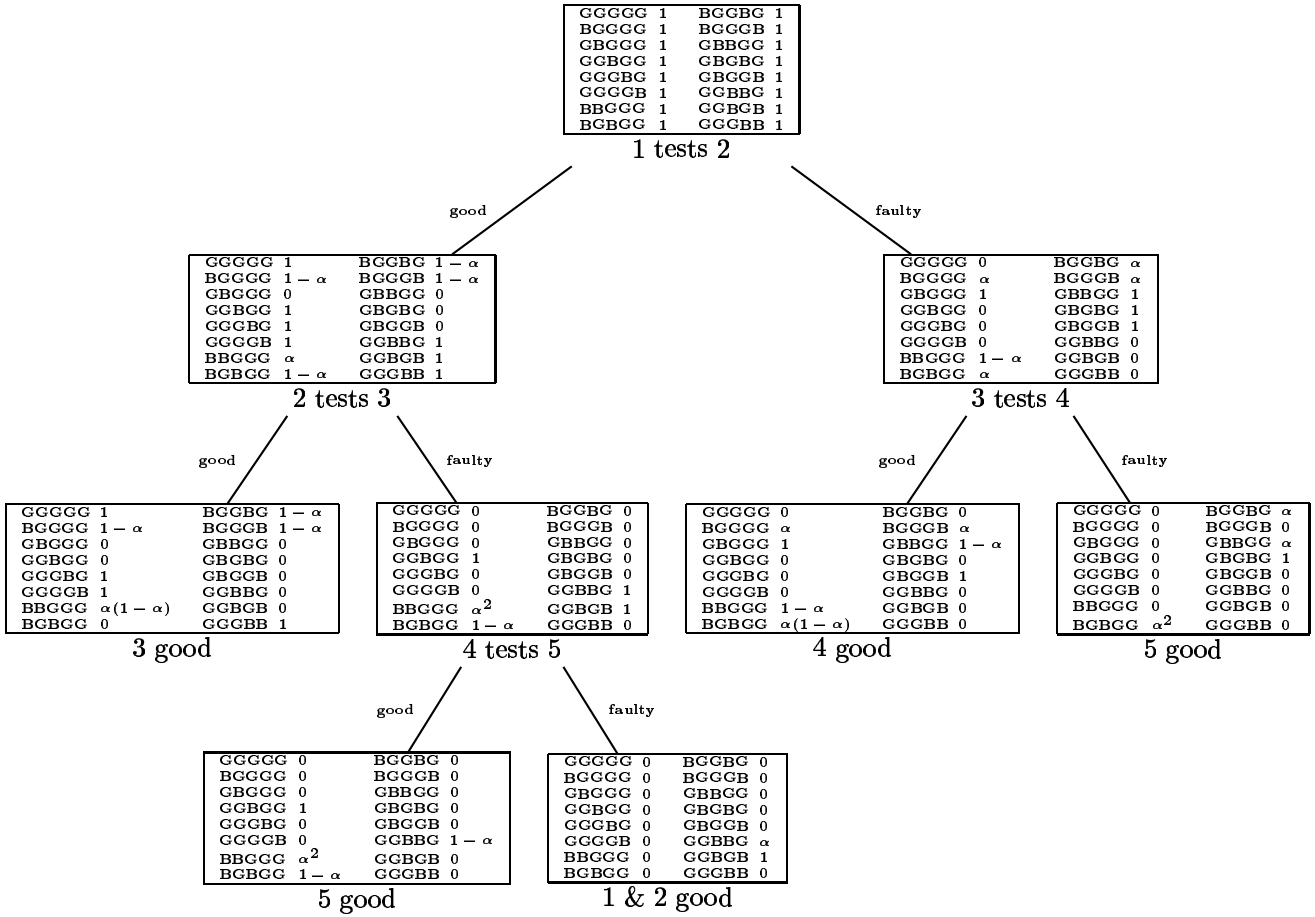

Figure 6: Optimal tree for $n = 5$ in the chip problem; in the unbiased case (if each of the 16 possible configurations is equally probable) its expected cost is $2.25 - (\alpha - \alpha^2)/16$. Each configuration is shown along with its frequency of occurrence, assuming a faulty chip gives a wrong answer with probability α.

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.

If there are any happy binomial trees left, let the largest one be of order k and have root R. 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 is bad, then all chips in R's subtree are 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. If R is bad, at least X_m chips are bad, where

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

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 [7] 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 m; the analysis depends on computation 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 [7] 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 $\mathbf{Pr}(X_m \leq n/2)$ exponentially small, but not to large, since the average cost increases linearly in m. When $\alpha = 1$ the choice

$$m_n(\rho, 1, \epsilon) = \frac{n}{2p} + n^{1/2+\epsilon},$$

for some ϵ, $0 < \epsilon < 1/2$, results from basic considerations about the binomial distribution (see [4, Section 9]), while when $\alpha < 1$ the choice of a value $m_n(\rho, \alpha, \epsilon)$ meeting these requirements is more subtle.

When $\alpha < 1$, the set of chips that would be bad if 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 $m_n(\rho, \alpha, \epsilon)$ to be smaller than $n/2p$. As a consequence, the average complexity *decreases* when $\alpha < 1$. This is surprising, because one would expect uncertainty in the diagnosis to cause trouble when $\alpha < 1$, but on the contrary, the average cost turns out to be smaller by $\Theta(n)$ for the chip

problem than for the majority problem. The example of $n = 5$ for the unbiased case (Figures 5 and 6) illustrates this strange phenomenon.

The choice of $m_n(\rho, \alpha, \epsilon)$ and the resulting computation of the average performance of $A_n(\rho, \alpha, \epsilon)$ are more intricate than for the majority problem. In preparation for the analysis, we define:

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

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

and

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

An easy recurrence gives

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

for $i > 0$; the definition of τ_i gives

$$1 - 2\rho_i \leq \tau_i < 1, \tag{5}$$

and hence

$$1 - \tau_i \leq \frac{2}{\alpha}(\alpha\rho)^{2^i}. \tag{6}$$

From the righthand side of (5) it follows that $\sigma < 1$. We also have $\sigma > 0$ as follows. Choose I such that for $k \geq I$, $(\alpha\rho)^{2^k} \leq \alpha/4$. By (6) we have

$$-\ln\tau_k \leq \ln\frac{1}{1 - \frac{2}{\alpha}(\alpha\rho)^{2^k}},$$

and so

$$\begin{aligned} -\ln\tau_I\tau_{I+1}\tau_{I+2}\cdots &\leq \sum_{k\geq I} \ln\frac{1}{1-\frac{2}{\alpha}(\alpha\rho)^{2^k}} \\ &\leq \frac{4}{\alpha}\sum_{k\geq I}(\alpha\rho)^{2^k}, \end{aligned}$$

by our choice of I because $2x + \ln(1-x)$ is 0 at $x = 0$ and is increasing for $0 \leq x \leq 1/2$, and hence

$$\ln\frac{1}{1-x} \leq 2x,$$

for $0 \leq x \leq 1/2$. Therefore

$$\tau_I\tau_{I+1}\tau_{I+2}\cdots > 0$$

for some I, and hence

$$\begin{aligned}\sigma &= \tau_0\tau_1\tau_2\cdots \\ &= (\tau_0\tau_1\tau_2\cdots\tau_{I-1})(\tau_I\tau_{I+1}\tau_{I+2}\cdots) \\ &> 0.\end{aligned}$$

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 $\lambda(\rho,1) = p$, $\mu(\rho,1) = 2p\Phi(\rho)$, and $\lambda(\rho,\alpha)$ and $\mu(\rho,\alpha)$ are both decreasing in α. Finally, define

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

Our sample size will be

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

which we explain 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 $\frac{n}{2\lambda(\rho,\alpha)}$. A plot of the fraction of chips sampled in $A_n(\rho,\alpha,\epsilon)$, for various values of α and ρ, is shown in Figure 7.

Now, let $c_n(\rho,\alpha,\epsilon)$ be the expected number of tests used by $A_n(\rho,\alpha,\epsilon)$ under the probabilistic assumptions described at the beginning of this section. We prove in the following discussion that

$$\lim_{\epsilon\to 0}\lim_{n\to\infty}\frac{c_n(\rho,\alpha,\epsilon)}{n} = \Psi(\rho,\alpha). \tag{7}$$

As a corollary, then,

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

Note that $\Psi(\rho,1) = \Phi(\rho)$; we conjecture that for $\alpha < 1$ and $\rho > 0$,

$$\Psi(\rho,\alpha) < \Phi(\rho). \tag{8}$$

Figure 8 provides very powerful numerical evidence for this conjecture. A consequence of this conjecture would be that

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

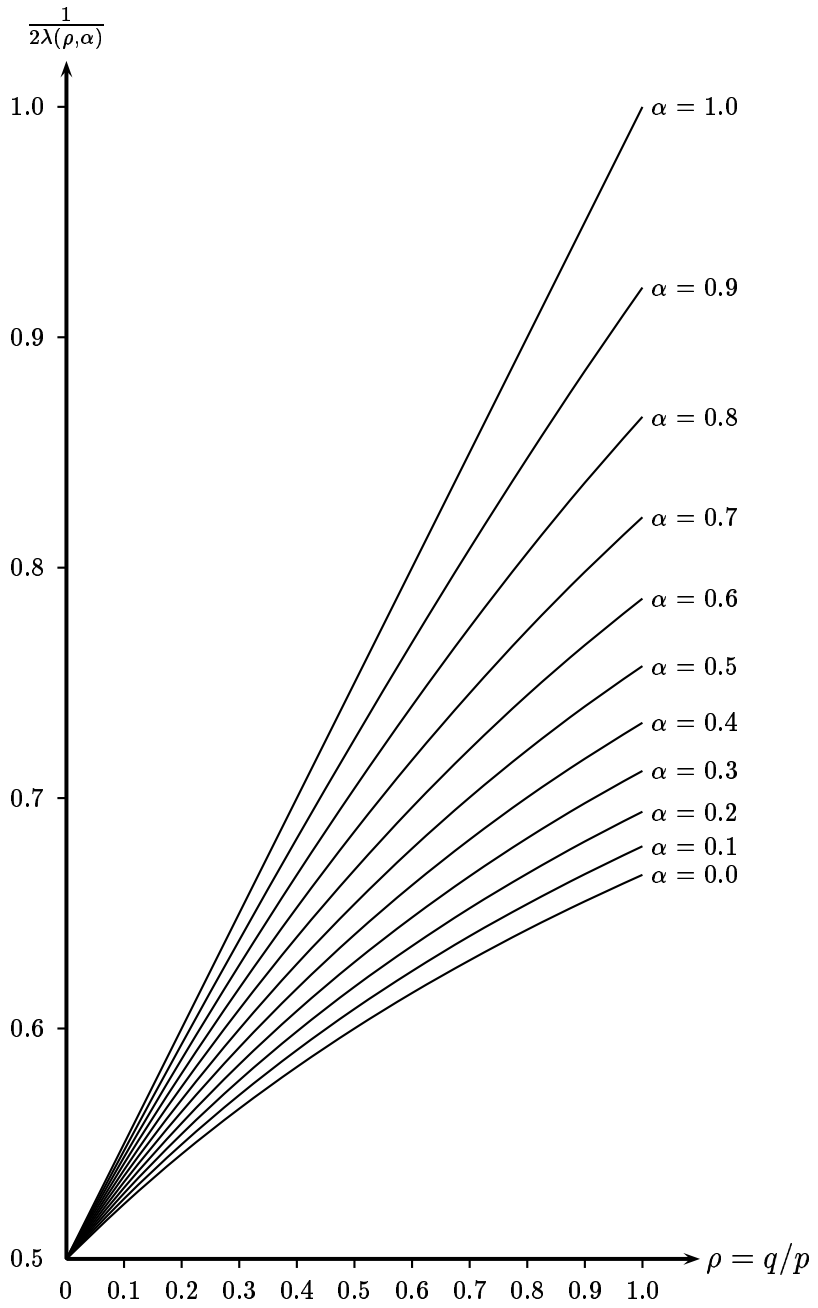

Figure 7: Plot of $\frac{1}{2\lambda(\rho,\alpha)}$, the fraction of chips sampled in $A_n(\rho, \alpha, \epsilon)$, for various values of α and ρ.

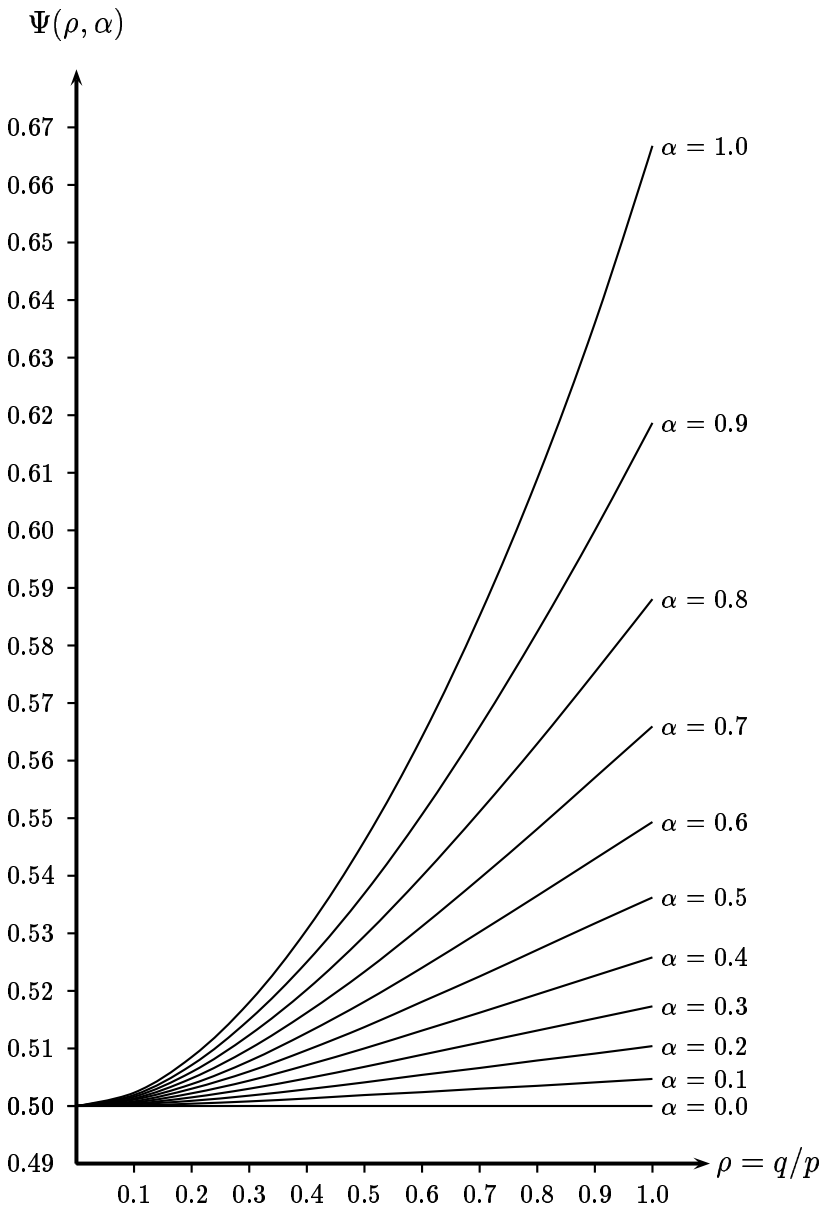

Figure 8: Plot of $\Psi(\rho, \alpha)$, the cost of $A_n(\rho, \alpha, \epsilon)$ per chip, for various values of α and ρ. We conjecture that for $\alpha < 1$ and $\rho > 0$, $\Psi(\rho, \alpha) < \Psi(\rho, 1)$.

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 that

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

Three observations support this conjecture. First, by our discussion of the choice of $m_n(\rho,\alpha,\epsilon)$, 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 [2] which is optimal in the worst case and within $O(\log n)$ of being optimal in the average case.

The analysis of $A_n(\rho,\alpha,\epsilon)$ rests on two results. First, for any $\epsilon > 0$ and any δ and γ satisfying $0 < \gamma < 1$, $0 < \delta < (1-\gamma)/2$,

$$\mathbf{Pr}(X_m < (\lambda(\rho,\alpha) - \epsilon)m) \le \frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta} + (2\delta \lg m) e^{-\sigma\tau_0 m^\gamma/4}; \qquad (9)$$

that is, it is exponentially small in a power of m. Second,

$$\lim_{m\to\infty} \frac{\mathbf{E}[T_m]}{m} = \mu(\rho,\alpha), \qquad (10)$$

where T_m is the number of tests performed by the algorithm.

With the sample size $m = m_n(\rho,\alpha,\epsilon)$, 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 $A_n(\rho,\alpha,\epsilon)$ thus satisfies

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

for any $\epsilon > 0$, and (7) follows.

In our distribution assumption at the beginning of this section, we lose independence, but this difficulty is surmounted by noting that for any random variable Z_n bounded by n, we have

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

so that

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

Taking absolute values,

$$\begin{aligned}&|\mathbf{E}[Z_n] - \mathbf{E}[Z_n \mid N > n/2]| \\ &\quad\le \mathbf{Pr}(N \le n/2)\left(|\mathbf{E}[Z_n \mid N > n/2]| + |\mathbf{E}[Z_n \mid N < n/2]|\right) \\ &\quad\le 2n\mathbf{Pr}(N \le n/2)\end{aligned}$$

where the expectation $\mathbf{E}[Z_n]$ is taken under the assumption of independent chip faultiness, and the expectation $\mathbf{E}[Z_n \mid N > n/2]$ according to the distribution $p^k q^{n-k}/\mathbf{Pr}(N > n/2)$. Now Chernoff's inequality [3, top of page 12] states that, if N is a binomially distributed random variable with parameters n and p, then

$$\mathbf{Pr}(|N - pn| \geq h) \leq 2e^{-2h^2/n},$$

so that

$$\mathbf{Pr}(N \leq pn - h) \leq 2e^{-2h^2/n} \tag{11}$$

because

$$\mathbf{Pr}(|N - pn| \geq h) = \mathbf{Pr}(N - pn \geq h) + \mathbf{Pr}(-N + pn \geq h)$$

and $\mathbf{Pr}(-N + pn \geq h) \geq 0$. Taking $h = (p - q)n/2$ in (11) we get,

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

and it follows that

$$|\mathbf{E}[Z_n] - \mathbf{E}[Z_n \mid N > n/2]| \leq 4ne^{-(p-q)^2 n/2},$$

so the difference is negligible and hence we assume independence of chip faultiness. With this assumption we can no longer insist that $N > n/2$; when $N \leq n/2$ algorithm $A_n(\rho, \alpha, \epsilon)$ is not guaranteed to find a good chip, but the algorithm is still well defined, as are X_m and T_m.

Let p_i and q_i be the probabilities that a chip at the root of a happy binomial tree $\mathcal{B}_i$ is good or bad; let $\rho_i = q_i/p_i$. Clearly $p_0 = p$, $q_0 = q$, and $\rho_0 = \rho$. We have the conditional probabilities

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

and

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

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,

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

establishing the inductive part of the definition of ρ_i. Step $i+1$ 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

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

Thus m_{i+1} is binomially distributed with parameters $\lfloor m_i/2 \rfloor$ and τ_i.

Because m_{i+1} is binomially distributed, Chernoff's inequality holds, so (11) tells us that for $i > 0$

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

Thus for any $t \geq 0$, $i > 0$,

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

since given that $m_i = 2s+t$, m_{i+1} is nondecreasing in t. Therefore for $i > 0$,

$$\begin{aligned}&\mathbf{Pr}(m_{i+1} \leq s\tau_i - h \text{ and } m_i \geq 2s) \\ &= \sum_{t=0}^{\infty} \mathbf{Pr}(m_{i+1} \leq s\tau_i - h \mid m_i = 2s+t)\mathbf{Pr}(m_i = 2s+t) \\ &\leq 2e^{-2h^2/s} \sum_{t=0}^{\infty} \mathbf{Pr}(m_i = 2s+t) \\ &= 2e^{-2h^2/s}\mathbf{Pr}(m_i \geq 2s) \\ &\leq 2e^{-2h^2/s}. \qquad (12)\end{aligned}$$

We now use (12) with a careful choice of s and h. Specifically, for any constant γ, $0 < \gamma < 1$, define

$$h_i = \begin{cases} \sqrt{m^{1+\gamma}} & i = 0, \\ h_{i-1}\tau_{i-1} & i > 0, \end{cases}$$

and

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

These are really functions of m, but we omit that from the notation for simplicity. By (12), for $i > 0$ we have

$$\begin{aligned}
&\mathbf{Pr}(m_i < s_i - h_i) \\
&\quad \le \mathbf{Pr}(m_i < s_i - h_i \text{ and } m_{i-1} \ge s_{i-1} - h_{i-1}) + \mathbf{Pr}(m_{i-1} < s_{i-1} - h_{i-1}) \\
&\quad \le 2\exp\left(\frac{-h_i^2}{4(s_{i-1} - h_{i-1})}\right) + \mathbf{Pr}(m_{i-1} < s_{i-1} - h_{i-1}), \\
&\quad \le 2e^{-\sigma\tau_0 m^\gamma/4} + \mathbf{Pr}(m_{i-1} < s_{i-1} - h_{i-1}), \qquad (13)
\end{aligned}$$

because

$$\begin{aligned}
\frac{h_i^2}{4(s_{i-1} - h_{i-1})} &\ge \frac{h_i^2}{4s_{i-1}} \\
&= \frac{m^{1+\gamma}(\tau_0\tau_1\tau_2\cdots\tau_{i-1})^2}{4m\frac{\tau_0}{2}\frac{\tau_1}{2}\cdots\frac{\tau_{i-2}}{2}} \\
&= m^\gamma 2^{i-3}(\tau_0\tau_1\tau_2\cdots\tau_{i-1})\tau_{i-1} \\
&\ge m^\gamma 2^{i-3}\sigma\tau_{i-1} \\
&\ge \frac{m^\gamma\sigma\tau_0}{4},
\end{aligned}$$

since the τ_i are increasing and $i > 0$. With the base case

$$\begin{aligned}
\mathbf{Pr}(m_0 < s_0 - h_0) &= \mathbf{Pr}(m < m - \sqrt{m^{1+\gamma}}) \\
&= 0,
\end{aligned}$$

a simple induction on (13) proves that

$$\mathbf{Pr}(m_i < s_i - h_i) \le 2ie^{-\sigma\tau_0 m^\gamma/4}. \qquad (14)$$

We will see below that $s_i = \mathbf{E}[m_i] + O(1)$, so this inequality says that m_i does not vary much from its expected value.

We can now prove (9) by applying (14) to a carefully chosen value of i. Inequality (14) holds for any $i \ge 1$, but it is of use to us only when $h_i = o(s_i)$, that is, when $\sqrt{m^{1+\gamma}} = o(m2^{-i})$, or equivalently, $2^i = o(\sqrt{m^{1-\gamma}})$. It suffices to take $i \le \delta \lg m$ for $0 < \delta < (1-\gamma)/2$. For such i, (14) becomes

$$\mathbf{Pr}(m_i < s_i - h_i) \le (2\delta \lg m)e^{-\sigma\tau_0 m^\gamma/4}. \qquad (15)$$

There are fewer than m_i tests in algorithm $A_n(\rho, \alpha, \epsilon)$ involving roots of happy trees of order at least i and hence size at least than 2^i—these are the tests performed at stages i, $i+1$, ..., together with some of the tests of the final stage. The probability of a "good" diagnosis is τ_i, so the probability of at least one "faulty" diagnosis is

$$\begin{aligned}
1 - \tau_i^{m_i} &= (1-\tau_i)(1 + \tau_i + \tau_i^2 + \cdots + \tau_i^{m_i - 1}) \\
&\le (1-\tau_i)m_i \\
&\le (1-\tau_i)m/2^i \\
&\le \frac{2}{\alpha}(\alpha\rho)^{2^i} m/2^i
\end{aligned}$$

by (6).

Thus 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" with a probability 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, 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,

$$\mathbf{Pr}\left(X_m < \frac{m + 2^i m_i - 2^i + 1}{2}\right) < \frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta}. \tag{16}$$

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

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

but the latter probability simplifies to yield

$$\begin{aligned}
&\mathbf{Pr}\left(X_m < \frac{m + 2^i(s_i - h_i) - 2^i + 1}{2}\right) \\
&\quad\leq \mathbf{Pr}\left(X_m \leq \frac{m + 2^i(s_i - h_i) - 2^i + 1}{2}\right) + \mathbf{Pr}(s_i - h_i < m_i) \\
&\quad\leq \frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta} + (2\delta \lg m) e^{-\sigma\tau_0 m^\gamma/4}
\end{aligned}$$

by (16) and (15). Taking the complement gives us

$$\begin{aligned}
&\mathbf{Pr}\left(X_m \geq \frac{m + 2^i(s_i - h_i) - 2^i + 1}{2}\right) \\
&\quad\geq 1 - \frac{2}{\alpha}(\alpha\rho)^{m^\delta} m^{1-\delta} - (2\delta \lg m) e^{-\sigma\tau_0 m^\gamma/4}. \qquad (17)
\end{aligned}$$

Now

$$\begin{aligned}
&\frac{m + 2^i(s_i - h_i) - 2^i + 1}{2} \\
&\quad= \frac{m(1 + \tau_0\tau_1\cdots\tau_{i-1})}{2} - \frac{2^i h_i + 2^i - 1}{2} \\
&\quad= m\left(\frac{1+\sigma}{2} - \frac{\tau_0\tau_1\cdots - \tau_0\tau_1\cdots\tau_{i-1}}{2}\right) - \frac{2^i h_i + 2^i - 1}{2} \\
&\quad= m\left(\lambda(\rho,\alpha) + \frac{\tau_0\tau_1\cdots\tau_{i-1}(1 - \tau_i\tau_{i+1}\cdots)}{2}\right) - \frac{2^i h_i + 2^i - 1}{2}. \qquad (18)
\end{aligned}$$

Elementary calculation yields

$$\begin{aligned}
\frac{\tau_0\tau_1\cdots\tau_{i-1}(1 - \tau_i\tau_{i+1}\cdots)}{2} &\leq \frac{1}{2}\tau_0(1 - \tau_i\tau_{i+1}\cdots) \\
&\leq \frac{1}{2}\tau_0[(1-\tau_i) + (1-\tau_{i+1}) + \cdots] \\
&\leq \frac{1}{2}\tau_0[2\rho_i + 2\rho_{i+1} + \cdots] \\
&= \tau_0[\rho_i + \rho_{i+1} + \cdots] \\
&\leq \frac{\tau_0}{\alpha}\sum_{j\geq 2^i}(\alpha\rho)^{2^j} \\
&\leq \frac{\tau_0}{\alpha(1-\alpha\rho)}(\alpha\rho)^{2^i} \\
&= O\left((\alpha\rho)^{m^\delta}\right) \\
&= o(1),
\end{aligned}$$

and

$$\begin{aligned}
2^i h_i + 2^i &= O(2^{\delta \lg m}\tau_0\tau_1\cdots\tau_{i-1}\sqrt{m^{1+\gamma}}) \\
&= O(m^{\delta + (1+\gamma)/2}) \\
&= o(m)
\end{aligned}$$

by our choice of δ and γ. Thus (18) becomes

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

for all $\epsilon > 0$ as m (that is, n) gets large, and (17) yields (9).

Finally, we prove (10). By (9), the likelihood of needing to apply the $\Theta(n)$ algorithm of [7] is exponentially small, so we ignore it in our estimates of T_m. Let $m_i = 0$ if $i > \lg m$. We have

$$\sum_{i=0}^{\lceil \lg m \rceil} \left\lfloor \frac{m_i}{2} \right\rfloor \leq T_m \leq \frac{1}{2} \sum_{i=0}^{\lceil \lg m \rceil} m_i + \lfloor \lg m \rfloor, \tag{19}$$

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. Given that m_{i+1} is distributed according to a binomial law with parameters $\lfloor \frac{m_i}{2} \rfloor$ and τ_i, for $i \geq 1$,

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

so,

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

Also,

$$\begin{aligned}\mathbf{E}[m_i] &\geq \frac{\tau_{i-1}}{2}\mathbf{E}[m_{i-1} - 1] \\ &\geq \tau_0\tau_1\cdots\tau_{i-1}m/2^i - \left(\frac{\tau_{i-1}}{2} + \frac{\tau_{i-1}\tau_{i-2}}{4} + \frac{\tau_{i-1}\tau_{i-2}\tau_{i-3}}{8} + \cdots + \frac{\tau_{i-1}\tau_{i-2}\cdots\tau_0}{2^i}\right) \\ &\geq s_i - 1.\end{aligned}$$

Thus

$$\mathbf{E}[m_i] = s_i + O(1),$$

and so,

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

By (19),

$$\sum_{i=0}^{\lceil \lg m \rceil} \mathbf{E}\left[\left\lfloor \frac{m_i}{2} \right\rfloor\right] \leq \mathbf{E}[T_m] \leq \frac{1}{2} \sum_{i=0}^{\lceil \lg m \rceil} \mathbf{E}(m_i) + \mathbf{E}(\lg m),$$

or,

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

and therefore

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

proving (10) as claimed.

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 obtained upper and lower bounds for the chip problem and have shown that the chip problem and the modified majority problem are the same in the worst case and 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 conjecture (8), 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.

How can we find all good chips, assuming any chip can test any other? Given a directed graph that tells us which chips can test which other chips, how can we find all good chips? What if the majority of the chips are bad: Can we identify a bad chip? A good chip? If bad chips always lie? If bad chips sometimes lie? For all of these problems, there are both the worst and average cases to be considered.

References

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

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

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

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

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

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

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

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

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

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

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

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