

# The Worst-Case Chip Problem\*

Laurent Alonso<sup>†</sup>   Philippe Chassaing<sup>‡</sup>   Edward M. Reingold<sup>§</sup>  
René Schott<sup>¶</sup>

September 4, 2003

**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. 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. 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.

**Key words.** Fault diagnosis, algorithm analysis, chip problem, majority problem

**AMS(MOS) subject classifications.** 68Q25, 68P10, 68Q20, 68M15

## 1 Introduction

In system diagnosis, according to [11], we consider

---

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

<sup>†</sup>INRIA-Lorraine and LORIA, Université Henri Poincaré-Nancy I, BP 239, 54506 Vandœuvre-lès-Nancy, France. Email: [Laurent.Alonso@loria.fr](mailto:Laurent.Alonso@loria.fr)

<sup>‡</sup>Institut Élie Cartan, Université Henri Poincaré-Nancy I, BP 239, 54506 Vandœuvre-lès-Nancy, France. Email: [Philippe.Chassaing@antares.iecn.u-nancy.fr](mailto:Philippe.Chassaing@antares.iecn.u-nancy.fr)

<sup>§</sup>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. Present address: Department of Computer Science, Illinois Institute of Technology, Stuart Building, 10 West 31st Street, Suite 236, Chicago, Illinois 60616-2987 USA. Email: [reingold@iit.edu](mailto:reingold@iit.edu)

<sup>¶</sup>Institut Élie Cartan and LORIA, Université Henri Poincaré-Nancy I, BP 239, 54506 Vandœuvre-lès-Nancy, France. Email: [Rene.Schott@loria.fr](mailto:Rene.Schott@loria.fr)

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.,  $u_i \in U$  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 [5, 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 [9]. 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.

When faulty chips always give wrong answers, the problem is reminiscent of the Smullyan’s puzzles [12] 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. Blecher [4] discusses a similar puzzle when liars sometimes tell the truth.

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 [8] give an excellent summary with a useful bibliography. 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 the worst 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 we discuss the worst-case of the chip problem in two forms: when bad chips always lie and when bad chips sometimes lie; both problems turn out to be equivalent to the majority problem.

## 2 Majority Problems

The *majority problem* of [1], [2], and [10] is to determine the majority color in a set of  $n$  elements  $\{x_1, x_2, \dots, x_n\}$ , each element of which is colored either blue or red, by pairwise equal/not equal color comparisons. When  $n$  is even, we must 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 [10]; [1] gave a short, elementary proof; [13] found yet a different approach. [2] proved 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.

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 Faulty Chips Always Lie

When faulty chips *always* lie, the chip problem is equivalent to the modified majority problem; we prove

**THEOREM 1.** *The problem of determining a good chip when faulty chips always give wrong answers is equivalent to the modified majority problem described above.*

*Proof.* The equivalence is based on 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 equivalent to the modified majority problem.  $\square$

## 4 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, surprisingly, these two problems are *equivalent* in the worst case:

**THEOREM 2.** *Any algorithm that determines the majority color in the modified majority problem can be used to solve the chip problem.*

*Proof.* 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

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

each  $\Delta_i$  is the excess of one color over the other in subsets  $A_i, B_i$  where  $A_1 \cup B_1 \cup A_2 \cup B_2 \cup \dots \cup A_m \cup B_m$  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  $(\Delta_1, \Delta_2, \dots, \Delta_m)$ , 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  $\Delta_i$  and  $\Delta_j$  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  $\Delta$ -vector one shorter in length than its own  $\Delta$ -vector or its left child with a  $\Delta$ -vector one shorter in length than its own and its right child with a  $\Delta$ -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  $\Delta$ -vector contains enough information to determine the

majority. Since there must be a strict majority, we have

$$\Delta_1 > \sum_{i \geq 2} \Delta_i, \quad (1)$$

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  $S_1, S_2, \dots, S_m$  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*  $\Delta_i$ . 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*  $\Delta_i$ . Letting  $G(S)$  and  $B(S)$  be the number of good and bad chips in  $S$ , respectively, we have

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

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

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

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

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

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

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

Initially, the  $\Delta$ -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  $\Delta$ -vector  $(\Delta_1, \Delta_2, \dots, \Delta_m)$ . 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  $\Delta_i$  and  $\Delta_j$  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  $\Delta_i$  and  $\Delta_j$  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  $\Delta_i$  and  $\Delta_j$ . 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 1 depicts the interpretation just described.

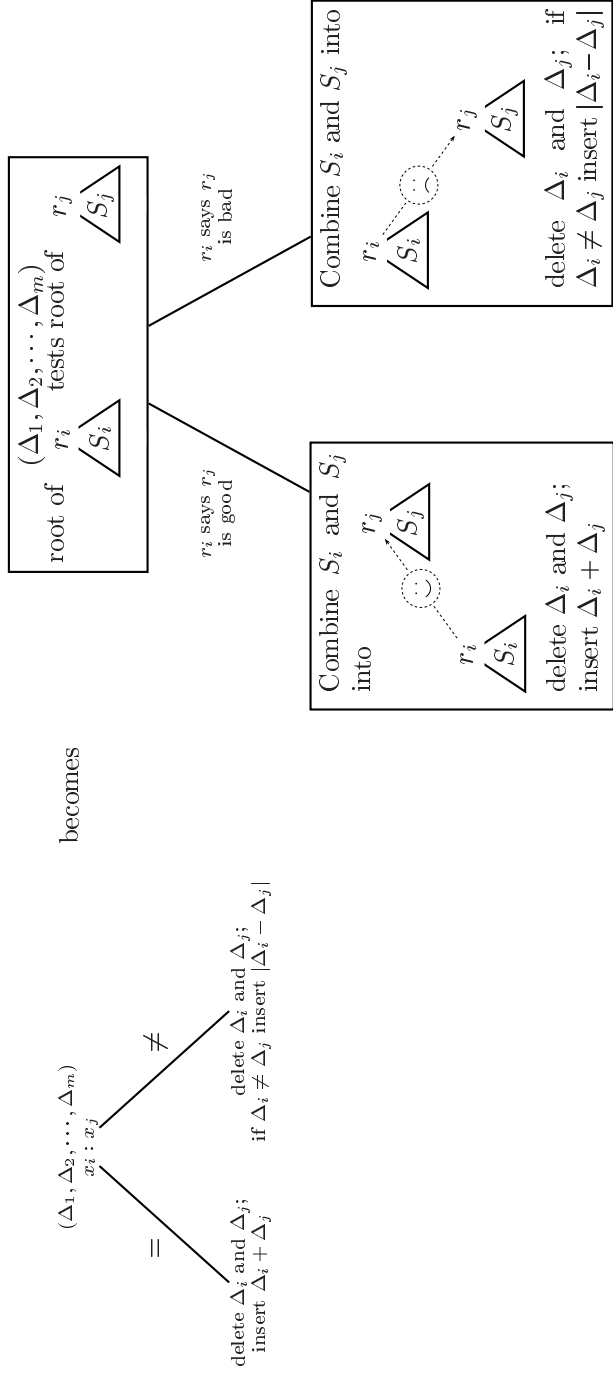


Figure 1: Interpretation of a decision tree for the modified majority problem as an algorithm for the chip problem.

First, consider the left child of the node (see Figure 1); 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 1); 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.  $\square$

## 5 Conclusions and Open Problems

We have shown that the chip problem and the modified majority problem are the same in the worst case. There is strong evidence that they are different in the average case [3].

## 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] Alonso, L., P. Chassaing, E. M. Reingold, and R. Schott, "The Average-Case Chip Problem," submitted for publication.
- [4] Blecher, P. M., "On a logical problem," *Discrete Math.* **43** (1983) 107–110.
- [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] Pelc, A. and E. Upfal “Reliable fault diagnosis with few tests,” *Combin. Probab. Comput.* **7** (1998), no. 3, 323–333.
- [9] Preparata, F., G. Metze, and R. T. Chien, “On the connection assignment problem for diagnosable systems,” *IEEE Trans. Comput.* **16** (1967) 848–854.
- [10] Saks, M. E. and M. Werman, “On computing majority by comparisons,” *Combinatorica* **11** (1991) 383–387.
- [11] Schmeichel, E. F., S. L. Hakimi, M. Otsuka, and G. Sullivan “A parallel fault identification algorithm,” *J. Algorithms* **11** (1990) 231–241.
- [12] Smullyan, R., *What Is the Name of This Book?—The Riddle of Dracula and Other Logical Puzzles*, Prentice Hall, Englewood Cliffs, New Jersey, 1978.
- [13] Weiner, G., “Search for a majority element,” *J. Statistical Planning and Inference* **100** (2002), 313–318.