Group Testing and Search

James Henry O'Geran

Submitted as fulfillment for the degree of Ph.D.

City University

Engineering Design Centre

May 1993

Contents

1 Introduction **12**

I GROUP TESTING AS A SEARCH PROBLEM **19**

2 Search **20**

2.1 Search 20

3 Related Topics **28**

3.1 Weighing 29

3.2 Multiaccess Communications 30

3.3 Search Trees and Information Retrieval 33

3.4 Hashing and the Substring Test 36

3.5 Search Games 37

3.5.1 Twenty Questions 38

3.5.2 Mastermind 39

II BINARY GROUP TESTING 42

4 Binary Group Testing 43

5 Multi-Stage Stepwise Group Testing 51

5.1 The Stepwise Procedure 52

5.2 Three-Stage Stepwise Group Testing 57

5.3 s-Stage Stepwise Group Testing 58

5.4 Comparison of Stepwise Algorithms 60

5.5 Approximation and Upperbound to $R(s)$ 61

5.6 A Bifurcation Algorithm 71

6 An Investigation of Non-Mixing Binary Group Testing Algorithms 75

6.1 Non-Mixing Group Testing Algorithms 76

6.1.1 Two-Stage Pooling 77

6.1.2 Multi-Stage Pooling 78

6.1.3 Two-Stage Stepwise Group Testing 79

6.1.4 Multi-Stage Stepwise Group Testing 79

6.1.5 S & G Bifurcation 80

6.1.6 Maximum Entropy 80

6.1.7 Optimal Algorithm 81

6.2 Expected Number of Tests 82

6.3 Probability Distribution Function 84

6.4 Variance 93

6.5 Worst Case Number of tests 99

6.6 Sensitivity Analysis 101

III ADDITIVE GROUP TESTING 105

7 Additive Group Testing 106

7.1 Multi-Stage Pooling Algorithms for Additive Group Testing 109

IV FACTOR SCREENING 115

8 Factor Screening with Unknown Error Variance 116

8.1 Split-Experiment Factor Screening 122

8.2 Unreplicated Factor Screening 133

8.3 Performance of Methods 136

V ROBUST CIRCUIT DESIGN — AN EXAMPLE 140

9 An Application of Group Testing to Robust Circuit Design 141

9.1 The Circuit 143

9.2 Group Testing Experiment 146

9.3 Conclusion 147

List of Tables

5.1 Optimum group sizes and the expected number of tests per unit, $R(s)/k_1$, and corresponding approximation $\hat{R}(s)/k_1$, for a single 1st-order group. . . 61

5.2 Approximation to the optimum size of 1st-order groups for an s-stage stepwise group testing algorithm 70

5.3 Expected number of tests per unit for optimal bifurcation algorithms . . . 74

6.1 Expected number of tests of algorithms 83

6.2 Probability of requiring no more than c tests for a population of size 50 . . 92

6.3 Variance of the number of tests of algorithms 99

6.4 Worst case number of tests of algorithms 101

6.5 Expected number of tests to classify 100 units when p is incorrectly estimated 103

7.1 Group sizes and expected number of tests per unit for optimum pooling algorithms when the response is additive 113

7.2 Expected number of tests to classify 100 units using pooling algorithms and optimal algorithm 114

8.1 Optimum group sizes for split-experiment factor screening with $\alpha_1 = 0.05$. 128

8.2 Optimum group sizes for unreplicated factor screening using PSE with $\alpha_1 = 0.05$ 135

8.3 Results of simulation study of split-experiment factor screening. Number of factors ≈ 100, $\alpha_1 = \alpha_2 = 0.05$. 138

8.4 Results of simulation study of factor screening using PSE to estimate σ. Number of factors ≈ 100, $\alpha_1 = \alpha_2 = 0.05$. 139

9.1 Number of tests using two-stage pooling algorithm 147

List of Figures

2.1 A sequential search algorithm 24

2.2 Optimal search tree for bad penny problem 26

3.1 Bifurcation algorithm for finding active terminals 31

3.2 A game of Mastermind 40

5.1 Flow chart defining the stepwise procedure for classifying a population of size k 53

6.1 Pdf of 2-stage pooling algorithm 87

6.2 Pdf of 3-stage pooling algorithm 87

6.3 Pdf of 4-stage pooling algorithm 87

6.4 Pdf of 2-stage stepwise algorithm 88

6.5 Pdf of 3-stage stepwise algorithm 88

6.6 Pdf of 4-stage stepwise algorithm 88

6.7 Pdf of stepwise bifurcation 89

6.8 Pdf of S & G bifurcation algorithm 89

6.9 Pdf of entropy algorithm 89

6.10 Pdf of optimal algorithm 90

6.11 Sensitivity plot of 2-stage group testing algorithm 102

6.12 Sensitivity plot of 3-stage group testing algorithm 102

6.13 Sensitivity plot of 4-stage group testing algorithm 102

6.14 Sensitivity plot of 2-stage stepwise algorithm 102

6.15 Sensitivity plot of 3-stage stepwise algorithm 102

6.16 Sensitivity plot of 4-stage stepwise algorithm 102

6.17 Sensitivity plot of stepwise bifurcation algorithm 102

6.18 Sensitivity plot of entropy algorithm 102

6.19 Sensitivity plot of optimal algorithm 102

9.1 Electronic circuit for controlling the joystick on an electric wheelchair . . . 144

9.2 Main effects plots of components 145

9.3 Main effects plots of group-factors with k=3 149

9.4 Main effects plots of group-factors with k=5 149

9.5 Main effects plots of group-factors with k=7 150

9.6 Main effects plots of group-factors with k=9 150

Acknowledgement

I wish to thank Henry Wynn for his supervision and guidance throughout this work, and for his help and advice in many other matters. Also Anatoly Zhiglyavsky with whom many conversations greatly enhanced my understanding of group testing and search. I wish also to thank Alan Jebb for help in many practical matters, and Ron Bates for taking time out to run the simulations in the circuit example.

For the whole of this work I was funded by the Science and Engineering Research Council, U.K., firstly as a student in the Department of Actuarial Science and Statistics, and thereafter as a research assistant in the Engineering Design Centre.

I grant powers of discretion to the University Librarian to allow this thesis to be copied whole or in part without further reference to me. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

Abstract

Group testing is a method for detecting the small number of important units from a large population in situations where groups of units may be tested together to observe whether any members of the group are important. If a group is found not to contain any important units then it is classified immediately, otherwise its members are investigated further. Group testing is also a special case of a search problem.

The fundamental ideas of search are discussed and formulated and group testing is expressed in terms of search notation. Links are made to other areas of search similar in nature to group testing.

A new binary group testing algorithm, multi-stage stepwise group testing is introduced and investigated. A study of this and other algorithms is carried out in terms of criteria other than the expected number of tests.

The case of additive group testing is considered and a new multi-stage algorithm is introduced. The expected number of tests of this algorithm is derived and its properties are compared to existing algorithms.

The problem of factor screening is discussed. It is argued that the only model assumption which has not been previously satisfactorily considered is that the error variance must be known. Two methods for dealing with this case are proposed, and their relative properties are investigated in a simulation study.

An application of group testing is presented using robust circuit design as an example, providing useful insight into the performance of group testing and providing a new area of application.

Chapter 1

Introduction

In an experiment a common problem is to detect the members of a population which have some special characteristic of interest. The particular case considered here is where the population is large and the number of units of interest is small. Examples of this include detecting the defective units from a batch of manufactured output, detecting the members of a population with a certain disease, and detecting the factors in an experiment which have a non-negligible effect on some response.

The experiment is performed by sequentially performing tests on units within the population, gaining information about which ones may be important, and finally determining a classification of the population. The aim is to devise an experimental strategy which classifies the population using a small number of tests. If errors are present then it is also required to achieve a high probability of detecting any important unit and a low probability of incorrectly detecting any non-important unit. Other criteria such as ease of implementation and maximum number of tests per unit will also be considered when designing the experiment and the final choice of method will depend on the experimental

situation.

The distinguishing feature of a *group testing experiment* is that units may be formed into a group and a test performed on the whole group simultaneously (a so called *group test*). The observed outcome of such a test is related to the presence of important units in the group, for example a test may reveal the exact number, or simply the presence or absence of important units. In particular, a test is able to detect (or estimate) groups which contain no important units. Therefore, if the proportion of important units is small, then a good grouping scheme will often be able to classify a whole group of non-important units using a single test.

Example 1.1 Blood Testing

A classic group testing problem is that of blood testing, and it was in this context that group testing was first introduced by Dorfman (1943). During World War II it was necessary to test new recruits for syphilis. Testing recruits individually was costly and since most tests were negative, also wasteful. Dorfman proposed that blood samples from a group of recruits be pooled together and a single test performed on the pooled sample. If the test was negative then all recruits in the group were assumed not to have the disease, and thus a single test sufficed for the whole group. If the test was positive however then the recruits were tested individually. The aim therefore is to choose the size of each group in order to minimize the number of tests required. More sophisticated algorithms involve splitting units from groups tested positively into smaller subgroups, gradually isolating important units into smaller and smaller groups.

Example 1.2 Factor Screening

In the planning of an experiment a large number of factors may be proposed which could

possibly affect the response. In such a case conventional experimental designs may be prohibitively large. However, it may be known or suspected that only a few of the factors actually have any real effect. By simultaneously varying the levels of a group of factors and observing whether or not this has any significant effect on the response, group testing may be used to screen out the effective factors, and thus make subsequent experimentation more efficient.

It will be useful to define three categories of group testing, namely binary, additive and factor screening, and to classify group testing algorithms accordingly. The distinction between different categories depends on the response observed from group tests. Since different forms of response lead to a different interpretation of results, different approaches will be required for each case. This classification is considered useful since in previous studies authors have often made assumptions which were not subsequently used. In particular, the assumption of additivity has often been made which allows greater inferences to be drawn from tests than the binary case. However, since the intention of some authors was to provide a starting point for subsequent inclusion of errors, the additive property has often been ignored. Thus, as well as providing a framework for factor screening, such studies also describe useful methods for the binary case. By classifying methods according to the following definitions, any such ambiguities are overcome.

Case 1: Binary Group Testing

A binary group testing situation is where the outcome of a test on a group is binary, giving a negative result (or 0) if the group contains no important units and a positive result (or 1) if the group contains at least one important unit. In the event of a positive outcome it is not known which unit or units in the group are important, nor

how many are important. In binary group testing it is usual to consider the important units as defective and the non-important units as good, and this convention will be used in the chapter on binary group testing.

Case 2: Additive Group Testing

An additive group testing situation provides more information in a test than the binary case. In this case each unit has some value or *effect* associated with it. When testing a group of units the observed outcome is the sum of the effects of all units in the group. It is assumed that all non-important units have the same effect (usually taken to be zero) and that this value is known. Important units have some other effect and so can be detected by comparing the observed test result with that which would have been observed if all units in the group were non-important. Extra information is available for this case since, if a subgroup of a previously tested group is tested then the difference of the two outcomes gives the result for the complement of the subgroup as well. A special case of additive group testing is where the effects of important units are equal and known. In this case a test result reveals exactly how many important units are in the group. In additive group testing important units will be called *effective*.

Case 3: Factor Screening

In factor screening, as in Case 2, all units have an effect associated with them. In this case however, the outcome of a test is the sum of the effects of the units in the group plus some random error. This therefore includes the usual regression model and the aim is to detect the independent variables which have a non-negligible effect on the response variable. Although a generalization of Case 2, these methods are

best considered separately since considerations such as symmetry of designs and extra replication for error lead to substantially different approaches.

There have been many and varied published examples of where group testing methods have been directly applied in real problems. For example Hunter and Mezaki (1964) used a two-stage group testing method to select the best catalyst for a certain catalytic reaction, Smith (1973) used group testing methods in the design of a simulation 'optimizer', Thomas et al (1973) applied such methods to test large inventories of radioactive sources and Thompson (1962) used group testing to estimate the proportion of insects capable of spreading a virus in a population of six-spotted leafhoppers. A recent context in which blood testing has been used is in HIV testing, see for example Monzon et al (1992).

This thesis discusses group testing as a search problem and considers each of the three cases described above.

In Chapter 2 the general area of search is considered. Search notation is introduced and the fundamental ideas and concepts of search are discussed. In Chapter 3 links are made to other areas of search which are similar in nature to group testing, giving rise to interesting generalizations.

Binary group testing is described in Chapter 4. A new multi-stage stepwise algorithm is introduced in Chapter 5. The expected number of tests is derived and 2, 3 and 4-stage algorithms are compared. An asymptotic approximation to the expected number of tests is obtained for small p and minimized with respect to the group sizes in each stage. A new bifurcation algorithm is seen to be a special case and the properties of this algorithm are studied. Chapter 6 provides an investigation of many non-mixing algorithms. Previously only the expected number of tests has been considered and so this chapter provides deeper

insight into the relative properties of algorithms. Recursive equations are obtained for computing the p.d.f. of the number of tests which allow many properties of algorithms to be examined and compared. Expressions are also obtained for the variance and worst case number of tests and a sensitivity analysis is carried out. The general conclusions are that stepwise algorithms outperform pooling algorithms, entropy provides an excellent alternative to the optimal algorithm, and simpler algorithms are preferable if the number of defectives could be much higher than expected.

The case of additive group testing is examined in Chapter 7. A multi-stage version of the pooling algorithm is proposed and analyzed, and is seen to provide a simple and efficient alternative to the optimal power-of-two algorithm.

The problem of factor screening is considered in Chapter IV. It is argued that requiring that the error variance must be known is the main assumption preventing factor screening from being a fully practical method. Two methods for dealing with the case of unknown variance are proposed. The first method, split-experiment factor screening, involves performing two separate experiments in the first stage and swapping the estimates of the error variance in the second stage. Expressions for the expected number of tests and the average number of false classifications are obtained and the optimal group size is derived. The second method, unreplicated factor screening, uses results from the theory of unreplicated factorials to estimate the error variance. A simulation study shows that both methods can lead to efficient detection of effective factors using relatively few runs.

In Chapter 9 an application of group testing to robust circuit design is presented in which group testing is used to detect the important components in an electronic circuit where experimentation may be computationally expensive.

Throughout the group testing literature, with very few exceptions, algorithms have been developed to minimize the expected number of tests used and, where appropriate, to minimize the number of incorrect classifications. However, in certain circumstances other optimality criteria may be of greater interest, and this would appear to be a fruitful area for further research. For example, it may be desirable to have algorithms which do not test any particular unit more than a given number of times, or there may be a limit to the total number of tests allowed, and so one would wish to use an algorithm which has a high probability of classifying the whole population using no more than the allowed number of tests. Consideration may also be given to practical properties, for example ease of implementation, or speed of computation of the algorithm (an example of the latter is given in Section 3.4). Where several criteria are of importance, an algorithm may be developed in order to minimize some given numerical function of the criteria. Although no such algorithms have been developed in this thesis, Chapter 6 provides an investigation of group testing algorithms in terms of criteria other than the expected number of tests.

Part I

GROUP TESTING AS A SEARCH PROBLEM

Chapter 2

Search

Group testing may be thought of as an example of a search problem in which an unknown target is searched for by making observations at selected test sites. A review of search is given in O'Geran, Wynn and Zhiglyavsky (1991). In the following chapter other search problems which are similar in nature to group testing will be discussed, but first it will be instructive, and also useful to later work, to discuss the fundamental ideas and concepts of search as formulated and developed in [27], and to express group testing as a search problem.

2.1 Search

The basic idea of search is that there is some unknown *target set*, T, which the searcher tries to identify by observing the outcome of a *search function*, f, at selected *test sets*, $X_1, X_2, \ldots$. The *target field* $\mathcal{T}$ is the collection of all possible target sets which, in the finite case, may be written

$$\mathcal{T} = \{T_1, T_2, \ldots\}$$

so that $T = T_j$ for some j. Similarly the *test field* $\mathcal{X}$ is the collection of all test sets, so that each test is performed at a selected point in $\mathcal{X}$. The outcome of such a test is called an *observation* and depends on the unknown target and the selected test set. This is written $f(X,T)$ and assumed to be a real valued vector so that

$$f : \mathcal{X} \times \mathcal{T} \longrightarrow \Re^d$$

for some $d \geq 1$. A search problem is then defined by the triple

$$\mathcal{S} = (\mathcal{T}, \mathcal{X}, f).$$

A number of definitions summarize the basics of a search. The first definition concerns the ability to find a target.

Definition 2.1 A target set $T \in \mathcal{T}$ is said to be *achievable* within the search problem $\mathcal{S} = (\mathcal{T}, \mathcal{X}, f)$ if for any $T' \in \mathcal{T}$ such that

$$f(X,T) = f(X,T') \quad \text{for all} \quad X \in \mathcal{X}$$

it follows that $T = T'$.

The idea here is that if no other target set would give *exactly* the same outcome as the target, no matter where the observation was made, then the test field $\mathcal{X}$ together with the search function f is enough to *distinguish* the target T.

Now since T is unknown, the last definition leads naturally to

Definition 2.2 A search problem $\mathcal{S}$ is *solvable* if every T in $\mathcal{T}$ is achievable within $\mathcal{S}$.

Now for a search problem to be solvable it is required that for every pair of target sets in $\mathcal{T}$ there must be at least one test set in $\mathcal{X}$ which can distinguish between them. This leads to

Definition 2.3 A test set $X \in \mathcal{X}$ *separates* the target sets T_i and T_j if

$$f(X, T_i) \neq f(X, T_j).$$

The target is found when T is separated from all other $T_j \in \mathcal{T}, T_j \neq T$. Upperbounds for search algorithms may be obtained by randomly choosing the test sets and computing the probability of separating the target sets (O'Geran, Wynn and Zhiglyavsky (1992, 1993b) contain applications and developments of this idea).

For a solvable search problem then, it is always possible to find the target by observing f at every set X in $\mathcal{X}$. Naturally this would usually be inefficient and it is required to devise a strategy for choosing a selection of test sets which can find T without looking everywhere. An *algorithm* is a procedure which results in an ordered collection of test sets

$$\mathcal{A} = X_1, X_2, \ldots, X_L \subseteq \mathcal{X} \quad (L \leq \infty)$$

which determines what observations are made during a search. An algorithm is called *sequential* if the observations

$$Y_1 = f(X_1, T), Y_2 = f(X_2, T), \ldots, Y_t = f(X_t, T)$$

are used to determine the next test set X_{t+1}. Conversely an algorithm is *non-sequential* if the complete choice of sets to be tested is made before the experiment begins.

An algorithm solves the search problem $\mathcal{S}$ if the subsearch problem

$$\mathcal{S}^* = (\mathcal{T}, \mathcal{A}, f)$$

is *always* solvable. "Always" here acknowledges the fact that for a sequential algorithm $\mathcal{A}$ may depend on T.

It is clear therefore that by making an observation $f(X,T)$, the collection of possible target sets is reduced. Only those $T_j \in \mathcal{T}$ which would have given the observed outcome if they were the target remain candidates to be the target. As the search proceeds and more observations are made, the number of candidates gets smaller. Formally,

Definition 2.4 For an algorithm $\mathcal{A}$ and target T, a set $T_j \in \mathcal{T}$ is said to be *t-consistent* if

$$f(X_i, T) = f(X_i, T_j) \quad (i = 1, 2, \ldots, t).$$

It is clear that each test set X induces a partition of the target field $\mathcal{T}$ according to the possible outcomes of the observation $f(X,T)$. The partition is such that the T_j within blocks would all give the same outcome and those across blocks would give different outcomes. Each block of the partition therefore contains those which would be consistent if the corresponding outcome was observed. Since the target is always consistent and has already been separated from inconsistent sets, at each stage a good choice of test set is one which induces a fine partition of the current consistent set. This fineness may be

measured by some uncertainty criterion, such as entropy. An investigation of entropy and other search algorithms is found in O'Geran, Wynn and Zhiglyavsky (1993a).

As a search proceeds the state of knowledge or belief about the possible candidates for T becomes more precise. This is measured by a *belief function* μ_t on $\mathcal{B}(\mathcal{T})$ where $\mathcal{B}(\mathcal{T})$ denotes the set of all finite subsets and intersections of $\mathcal{T}$. Before the experiment begins μ_0 summarizes the prior beliefs, and this is updated after each observation. The updating procedure of a typical search algorithm is represented in Figure 2.1.

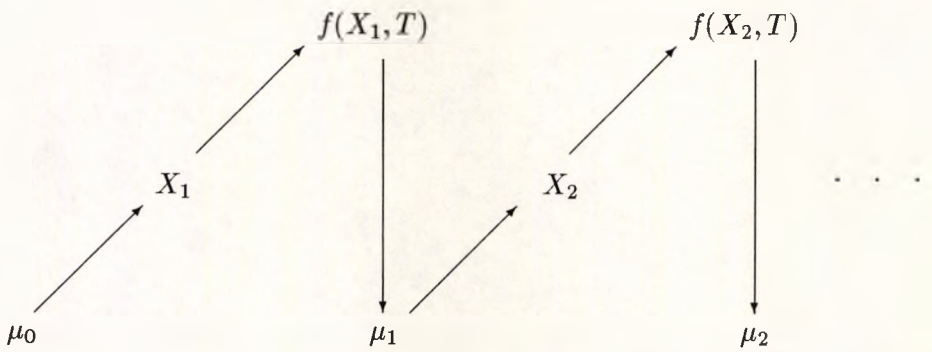

Figure 2.1: A sequential search algorithm

A common case is where μ_t is a probability measure giving the probability that any subset of $\mathcal{T}$ contains T. For example in group testing it is often assumed that each unit is independently important with probability p, so that μ_0 is induced by a function of independent Bernoulli random variables, and after each observation μ_t is updated using Bayes' theorem.

A convenient way to represent a sequential algorithm (and one which will be used for group testing in later chapters) is as a *decision tree.* In this context a tree consists of *inner nodes*, *branches* and *leaves.* Each inner node summarizes the current state of the search (for example in terms of μ_t or the consistent set), and stipulates the next test

to be performed. Descending from each inner node are branches corresponding to the possible outcomes of this test. Each branch leads either to a leaf if the search is finished or to another inner node if tests remain to be performed. The tree has a single inner node called the *root* from which all other nodes descend. This represents the first test to be performed. The search is defined by a directed path through the tree, performing the tests represented by each of the inner nodes encountered and following the branch corresponding to the outcome of the test until a branch leads to a leaf, at which point the search is finished. The leaf represents the subset of the target field $\mathcal{T}$ which is consistent with all of the test results observed along the path through the tree.

Example 2.1 Bad Penny

A number of coins are of the same weight except one which is heavier. Using a balance scale the odd coin is to be found using a minimum number of weighings. Figure 2.2 shows a tree representing the optimal solution for nine coins which are arbitrarily labeled $A, B, \ldots, I$. Using this algorithm the problem is solved using just two tests.

Now consider expressing group testing as a search problem. Suppose that the population consists of N units and let these be denoted by $x_1, x_2, \ldots, x_N$. The most general group testing situation is where any combination of units may be important. Let the target T be the set of important units. Then the target field is given by all subsets of the population set $\{x_1, x_2, \ldots, x_N\}$,

$$\mathcal{T} = \Big\{\emptyset, \{x_1\}, \ldots, \{x_N\}, \{x_1, x_2\}, \ldots, \{x_{N-1}, x_N\}, \ldots, \{x_1, x_2, \ldots, x_N\}\Big\}$$

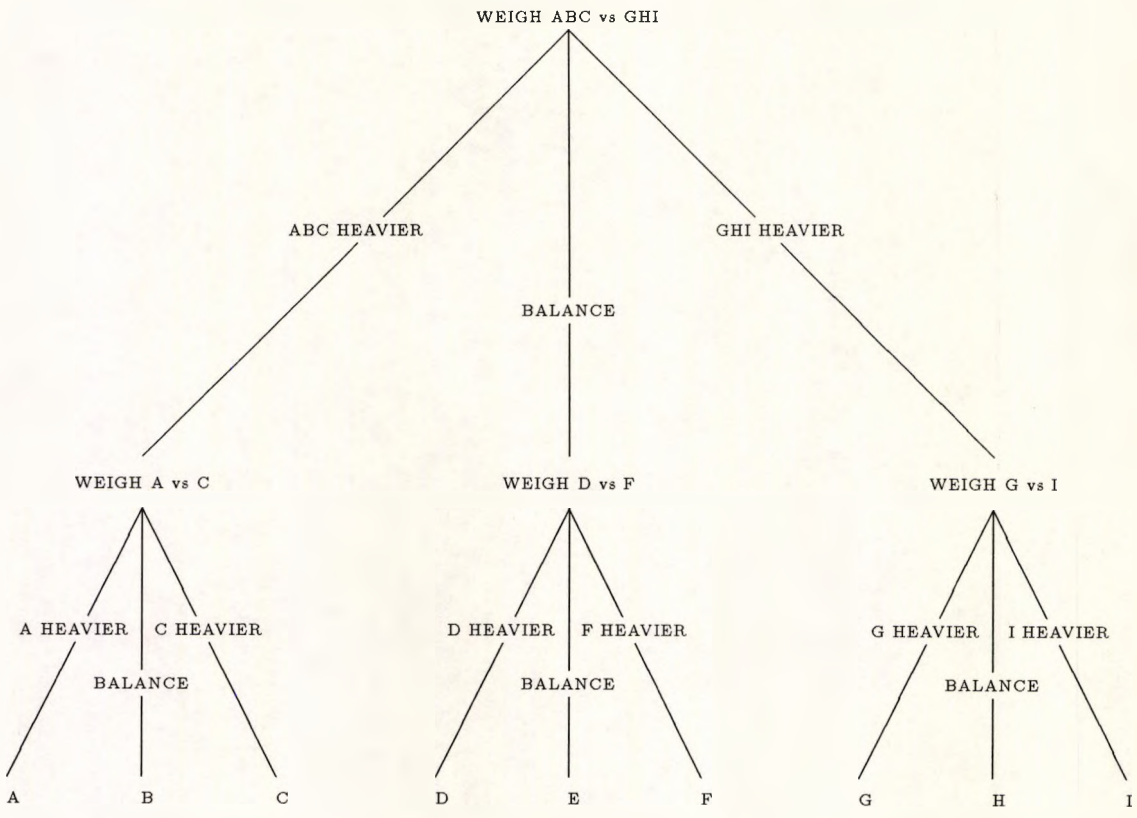

Figure 2.2: Optimal search tree for bad penny problem

(so for example the event $T = \{x_1, x_3, x_4\}$ corresponds to the units x_1, x_3, and x_4 being important and all other units non-important).

Similarly, in the most general case a test group may be any non-empty set of units. Thus the test field is the same as the target field minus the empty set

$$\mathcal{X} = \mathcal{T} - \emptyset.$$

Some group testing problems impose constraints on the target and/or test sets, for example the number of important units may be known beforehand, or there may be a limit to the number of units allowed in a test. In such cases it is straightforward to amend

$\mathcal{T}$ and/or $\mathcal{X}$ accordingly.

For binary group testing the response is whether or not the tested group contains any important units. In terms of the search function this may be given by

$$f(X,T) = \begin{cases} 1 & \text{if } X \cap T \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

In the additive case, let $\beta_1, \beta_2, \ldots, \beta_N$ be the effects of the units $x_1, x_2, \ldots, x_N$. Then the search function is given by

$$Y = f(X,T) = \sum_{i:x_i \in X} \beta_i.$$

Chapter 3

Related Topics

One of the main aims of the review paper [27] was to illustrate that the same ideas and problems are often found in separate areas of theory and research, and that by making the connection between these areas, under the combined subject of *search*, a more complete understanding of many aspects of theory is achieved, as well as the possible exchange of results and ideas, examples of which will be discussed.

The paper [27] provided an overview of search without details of specialized areas, and in particular the connection between group testing and other problems was only briefly covered. These connections will be more fully explored here, and new examples will also be covered. It will be seen that many of these problems can be considered as generalizations of group testing, based as they are on the same approach of sequentially testing groups of units in order to gain information about the units themselves. The differences between the experimental situations, however, give rise to challenging generalizations of group testing; for example constraints on the size of test groups, different optimality criteria, more complex test responses, and computability. How group testing ideas may be adapted

to these more general settings (or indeed how the ideas from these problems may be adapted to group testing) appears to be a fruitful area of further research. The following examples illustrate the wide application of group testing ideas within search theory.

3.1 Weighing

One problem whose relationship to group testing has long been recognized is weighing, an example of which was discussed in Example 2.1. A large batch of coins with equal, known weight is corrupted by a small number of coins of odd weight and it is required to find these odd ones.

There are two main variants of this problem. In the first variant a spring scale is used. By weighing a group of coins and comparing the result to that which would have been observed if no odd coins were present, the presence or absence of odd coins can be deduced (although care must be taken that two complementary odd coins weighed together do not cancel each other out). This problem is therefore identical to additive group testing.

The second variant of the problem uses a balance scale. This variant provides an interesting generalization of group testing in the case where it is known that there is only one odd coin (assume this to be heavier), as in Example 2.1. A test result can then lead to three different conclusions; if one or other group being weighed is heavier then that group contains the odd coin, otherwise the odd coin is amongst the coins not involved in that weighing. This therefore corresponds to a group testing situation with one important unit, where the population is split into three groups and a test reveals which group contains the important unit (there is of course the constraint that at least two of the groups must be of equal size). The problem becomes much more complicated however when two or more

odd coins are present.

3.2 Multiaccess Communications

An area in which group testing ideas developed independently is *multiaccess communications*. This problem involves a number of terminals which intermittently and independently wish to transmit information via some central processor which can only deal with one message at a time. The processor first determines which terminals are *active* (i.e. have a message to transmit), and then transmits their messages. One approach to the first of these functions, known as *polling*, is to query each terminal individually to discover whether or not it is active. However, Hayes (1978) pointed out that polling is very inefficient when only a small number of terminals are active, and that message delay is then more a function of the time required to poll all terminals than of heavy traffic. Hayes suggested that a more efficient method, which he called *probing*, would be to query groups of users simultaneously, the basic idea being to quickly eliminate groups of inactive terminals.

This problem reads very similarly to the blood testing problem of Example 1.1, and indeed represented a rediscovery of group testing with Hayes and subsequent authors developing algorithms very similar, and often identical, to those found in the group testing literature. This was not realised however, until some years and many papers later, after which Wolf (1985) dubbed multiaccess communications "born again group testing" in a paper whose purpose was "to give additional evidence to the well-known phenomenon that a good new idea is often the reincarnation of a good old idea." (although he also pointed out that in this case the "old idea" was not yet dead).

Figure 3.1 gives a representation of terminals connected to the central processor by a tree-like structure of transmitting wires. The central processor queries terminals by sending a signal which may originate from any junction of the tree, and which reaches all terminals which descend from that junction. The particular example shown in the figure represents the bifurcation algorithm of Hayes which begins by querying all terminals simultaneously, and if a positive response is received, then splits the population in half and repeats the query for each half, continuing recursively until all terminals have been classified as either active or inactive (this algorithm is in fact very similar to the binary group testing algorithm R_5 of Sobel and Groll (1959)).

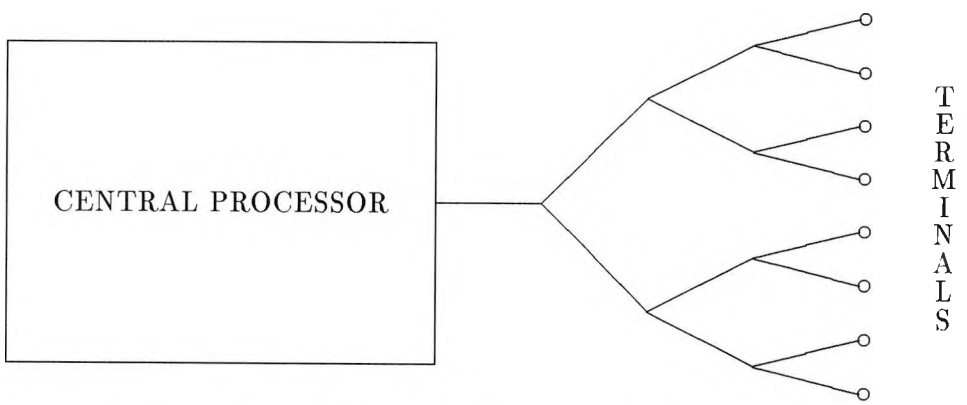

Figure 3.1: Bifurcation algorithm for finding active terminals

More efficient algorithms were developed by subsequent authors (see [45] for a review and list of references) until the connection to group testing was made by Berger et al (1984) who were thus enabled to draw upon results from the group testing literature, and in particular reported the optimal algorithm of Sobel and Groll (1959).

Multiaccess communications is therefore seen to be a very similar problem to group testing. However, there are a number of characteristics of multiaccess communications

which provide interesting generalizations of the basic group testing problem.

The first generalization is in the distribution of important units. In group testing it is generally assumed that units are independently important with some known or unknown probability. In multiaccess communications however, it is more commonly assumed that messages from each terminal arrive exponentially after the last one has been transmitted. This therefore adds a temporal aspect to group testing whereby the probability of a unit being important depends on the time since the last experiment was performed.

Another interesting generalization arises from the fact that the terminals have busy and quiet periods so that the rate at which messages are transmitted varies with time. An algorithm is therefore required to be adaptable to perform robustly as a function of the probability of units being important. In group testing it would be possible to vary the grouping scheme at different times. In multiaccess communications however, there is the strategic difficulty that the grouping is determined by the wired connection of terminals and is therefore fixed. Hayes suggested modifying algorithms to start at a lower level of grouping; so for example in the bifurcation algorithm of Figure 3.1, the central processor could consider the tree of eight terminals as two trees of four terminals and deal with them both separately. In very busy periods individual testing would also be permitted. This approach of splitting the population into groups which are then bifurcated will in fact be seen to be similar to a special case of multi-stage stepwise group testing which will be introduced in Chapter 5.

The multiaccess communications model described above where active terminals are first identified and their messages then transmitted is known as a *reservation protocol.* An alternative model, called a *direct transmission protocol* provides another generalization

of group testing. In this case, when an active terminal is queried it responds with the message itself. If this message arrives at the central processor alone then it is transmitted immediately, even if other terminals remain to be classified. When querying a group of terminals therefore, there are three possible outcomes. If no terminals in the group are active then the whole group is classified and need not be considered further. If exactly one terminal is active then its message is transmitted and again the group is classified and need not be considered further. If however more than one terminal is active then a collision of messages occurs and the group must be split into subgroups and requeried. This therefore gives a three-valued response function which indicates whether the group contains zero, one, or more than one important unit. Berger et al (1984) adapted the recursive equations of the optimal algorithm R_1 from [38] for this alternative response function. How other group testing algorithms may be similarly adapted provides another interesting research question.

3.3 Search Trees and Information Retrieval

An important problem which arises in many contexts is to search for a particular member of a population in which the units have some linear ordering, e.g. alphabetic or numeric. This problem has become especially common in computer science where it is often required to retrieve some piece of information from the computer's memory. There is a vast amount of literature on this subject and research is still very much active; many algorithms based on tree representations have been developed and their properties investigated. Although no attempt has been made to review this literature here, it would seem that many results are to be found in this area which could provide interesting and important developments to

group testing. Two examples of this are Hwang (1976) in which the optimum non-mixing binary group testing algorithm was shown to be a succession of alphabetic tree searches, and Hwang, Pfeifer and Enis (1981) where the optimum additive group testing algorithm was shown to be given by a search tree constructed using the power-of-two rule. It would therefore seem beneficial to consider group testing in terms of search trees. Hwang and Mallows (1979) laid the foundation for this by proving reliability theorems which specify which partitions of the target field are feasible in group testing, and thus specifying the constraints imposed by group testing on the construction of tree-like algorithms.

An *alphabetic tree search* will now be discussed and the connections to group testing highlighted. In terms of search notation, the problem is as follows. Suppose the target sets $T_i \in \mathcal{T}$ $(i = 1, 2, \ldots, M)$ (and hence the set of leaves on the tree) have some linear ordering such that $T_1 < T_2 < \cdots < T_M$. An alphabetic tree search is where the search function can be written in the form

$$f(X,T) = \begin{cases} -1 & \text{if } X < T, \\ 0 & \text{if } X = T, \\ 1 & \text{if } X > T. \end{cases} \tag{3.1}$$

This therefore corresponds to, for example, searching for an entry in an alphabetically ordered list of files or documents. There are two main versions of such a search which may be thought of as retrieval and insertion.

When retrieving a file, assuming that the file exists, a file is chosen to be 'tested'. If this file is the target file then the search is finished, otherwise it is observed whether the target file lies to the left or to the right and another test file is chosen accordingly. In this

case the test field and the target field are the same, i.e. $\mathcal{X} = \mathcal{T}$, and all three outcomes of $f(X,T)$ given in (3.1) are possible. This can be thought of as a group testing problem with a single important unit in which the population is partitioned into three groups, one of which contains only one unit, and it is observed which of these groups contains the important one.

When inserting a file however, assuming that the file does not have the same name as any other file in the list, the target corresponds to a 'space' between two files where the new file is to be inserted. Suppose there are $M-1$ files in the list. As before, the test field is the set of files; label these $X_1, X_2, \ldots, X_{M-1}$. Then the target field may be written as $\{T_1, T_2, \ldots, T_M\}$ where

$$T = \begin{cases} T_1 & \text{if } T < X_1, \\ T_j & \text{if } X_{j-1} < T < X_j, \\ T_M & \text{if } T > X_{M-1}. \end{cases}$$

Hence, in this case, $f(X,T)$ as in (3.1) can only take the values ± 1.

Insertion corresponds to a binary group testing problem in which the units within a population have some arbitrary ordering and it is required to search for the first important unit (this correspondence was in fact used in [18] to obtain the optimum non-mixing group testing algorithm). Consider performing a test on the first k units (with respect to the ordering). Then if the result of this test is negative (no important units in group), then the first important unit clearly lies to the right of some imaginary point between the kth and $(k+1)$th unit. If however a positive test outcome is observed (at least one important unit in group), then the first important unit lies to the left of this point.

See Knuth (1973) for further reading on computer search and search trees.

3.4 Hashing and the Substring Test

A problem which arises frequently in computer programming is to determine whether or not a string contains some specified substring. For example, many libraries have interactive computer programs which allow the user to specify a keyword and which then finds all books in the library which have that keyword in the title. Since in applications such as this the database is very large, fast routines are required for implementing the substring test.

One method which has been widely implemented, and which is similar in nature to group testing, is *hashing* which was first discussed in this context by Harrison (1971). The method is of use in situations where the substring test is likely to be unsuccessful, i.e. the string being tested is not likely to contain the specified substring. So for example, suppose it is required to find all books in a large list of titles which contain the word "search". Clearly only a small proportion of the total list will be selected and so it would therefore be worthwhile to perform a preliminary computationally faster test for necessary but not sufficient conditions that the substring be found. Hashing provides such a test. The method makes use of the ability to do many Boolean operations in parallel on a computer by representing strings by a binary signature. The term *hashing* arises from the fact that the signatures are not unique to any particular string. Harrison's technique is as follows.

A string S is represented by a binary signature $b_1 b_2 \ldots b_m$ in which a value of 1 for b_i indicates that S contains at least one element of the set E_i. It is clear that if S_1 is a substring of S_2 then the binary signature of S_2 will have ones in all positions that S_1 has ones. Therefore the signatures of S_1 and S_2 are compared, and if this condition is not satisfied then it is concluded that S_1 is not contained in S_2, otherwise S_1 and S_2 are

compared character by character.

Information about the order of characters in the string may be included in the signature by letting the sets E_i $(i = 1, 2, \ldots, m)$ correspond to ordered groups of characters.

The comparison to group testing is clear. Define a substring to be important if it is contained in some string S. Then considering the binary signature of a substring to represent the group of all strings with that signature, the signature test corresponds to testing to see whether any member of this group is important. If not then clearly the substring of interest cannot be important. Therefore, as with group testing, the method involves classifying units as unimportant by observing that a group containing those units is unimportant. If the group is declared important then closer inspection of the units is required. There is however an interesting difference in that, whereas in group testing it is required to find all important members in the group, here it is only required to determine whether a particular member of the group is important. There is also some difference in the aim of an experiment since the overall aim is to reduce computing time. If therefore considering using a more sophisticated grouping scheme, the computational time is of greater importance than the number of tests performed.

The technique of hashing is also used for information retrieval and thus provides an alternative to the methods discussed in Section 3.3. See [22] for further reading on this.

3.5 Search Games

Many popular games can be considered as examples of search problems, for example hangman where one aims to discover a hidden word by guessing letters contained in the word, and battleships where one searches for the enemy's ships and submarines. Indeed,

games are often used for the development and testing of algorithms in computer science and artificial intelligence (the '8 puzzle' is the classic test bed for any new A.I. algorithm). Two search games with similar characteristics to group testing will now be discussed.

3.5.1 Twenty Questions

Consider first the game of twenty questions in which one player writes down the name of a famous person and the other player then tries to discover the identity of this person by asking a maximum of twenty questions which can only be answered yes or no.

The analogy to group testing is made by considering the target person to be the single important unit in a population. Each question can then be considered as testing a group with a binary response. So for example, "Is this person a politician" is equivalent to performing a test on the group of all politicians to see whether or not that group contains the important unit.

This game, while perhaps of no great mathematical interest, illustrates that search is a natural, intuitive process. As a search problem it is clearly ill-defined since the target field cannot be precisely specified (what exactly defines a "famous person"?) and the test groups permissible are dictated by the range of questions which may be sensibly asked, and yet a typical game contains many elements of a good search procedure, asking very general questions at the beginning and gradually homing in on the target as more information is obtained and the questions become more precise. Clearly no player decides beforehand to employ, for example, a maximum entropy algorithm, and yet players tend to ask questions which obtain maximum information (the common opening question "Is it male or female?" comes perhaps as close as possible to maximum entropy which in this

case is bifurcation). Although equally ill-defined, an updated belief function also plays an important part. A player will often venture a particular name as a question at an early stage in the game when the consistent set of candidates may still be large. This would suggest some non-uniform belief function on the target field which is vital to the game and is yet unspecifiable.

Although twenty questions as discussed above is ill-defined as a search problem, a more formal version of the game has appeared in applied mathematics literature. In this game the target is a number between one and one million (just less than 2^{20}), and is played by asking "is the number in the group G?". However, it is assumed that the player with the secret number is allowed to lie a fixed number of times. For further reading on this problem see [7], [15] and [43]. An interesting connection to group testing arises here since "lying" in this sense is similar to binary group testing with errors where it is assumed that observations occasionally give a false indication of whether or not the group contains defectives, see for example Graff and Roeloffs (1974).

3.5.2 Mastermind

A more formally defined search game which has similarities to group testing is *Mastermind*, also known as *Bulls and Cows*. This is a code-breaking game for two players in which the first player conceals a "target" which is a code of digits (or coloured pegs), and the second player tries to discover the code by sequentially presenting test codes, for each of which a score is received. The score consists of the number of "bulls" and "cows", where a bull is achieved for each digit correct and in the correct position, and a cow for each digit which is correct but in the wrong position. The game continues until the second player

determines (often with considerable thought) the target code. Figure 3.2 shows a simple game in which the target is a four-digit code chosen from the digits 0 to 6. The first line is the target, T, unknown to the second player. Lines 2 to 6 represent the second player's trials X_1 to X_5, and on the right are the bull and cow scores achieved by these trials. So for example, on the third trial the test code (2354) and the target code (2416) have two digits in common, one in the correct position (the 2) and one in the wrong position (the 4), and hence the score is one bull and one cow. The target is found using five trials.

T	2	4	1	6	b	c
X_1	0	1	2	3	0	2
X_2	1	0	4	5	0	2
X_3	2	3	5	4	1	1
X_4	2	4	0	6	3	0
X_5	2	4	1	6	4	0

Figure 3.2: A game of Mastermind

In general, the target consists of m digits chosen from a set of d, $T = (t_1, \ldots, t_m)$ say, where $t_i \in 0, \ldots, d-1$ (in the example of Figure 3.2, $m = 4$, $d = 7$). If $m = 1$ the game is trivial; hence it is assumed that $2 \leq m \leq d$. There are two variants of the game. In the "eastern" variant, components within the target T and the test sets $X_1, X_2, \ldots$ must be distinct, whereas in the "western" variant components may be repeated.

Mastermind can be considered as a group testing problem in which the population is of size d and the number of important units and the size of test sets are both fixed to be m. The game provides an interesting generalization of group testing however in that

the important units are ordered and it is required to discover not only which units are important, but also their positions. The search function, which in this case is the number of bulls and cows, can then be thought of as a group test which reveals exactly how many important units are in the test group, and also gives some partial information on which ones are important.

A study of Mastermind appears in [29] in which a number of search algorithms were applied to the game. Mastermind was also used as a test bed for new upperbounds obtained using randomization in [28] and [30].

Part II

BINARY GROUP TESTING

Chapter 4

Binary Group Testing

A binary group testing problem is most commonly expressed as a search for the small number of defective units within a large population. The distinguishing feature of binary group testing is that a group of units may be tested together to discover whether or not the group contains any defectives. It is not observed which, if any, are defective, nor how many defectives there are. Therefore, denoting a positive outcome by '1' and a negative outcome by '0',

$$\text{Test result} = \begin{cases} 0 & \text{if group contains no defectives} \\ 1 & \text{otherwise} \end{cases}$$

Definition 4.1 A group of units is called a *defective group* if the following two conditions hold,

i. the group is known to contain at least one defective,

ii. the outcome of a test performed on any proper subset of the group cannot be exactly predicted from the previous test results.

Condition (ii) simply imposes that a defective group is of minimal size, i.e. it cannot be reduced to a smaller group also known to contain at least one defective. There are two ways that a defective group can arise, either by testing a group of units with a positive outcome, or by the removal of non-defective units from a defective group.

Typical examples of binary group testing problems include electrical devices connected in series ("the Christmas tree lighting problem") where any defective devices in the series will cause the whole system to fail thus indicating the presence of at least one defective, the detection of leaking cylinders where a number of cylinders are placed in an airtight chamber and the presence of at least one leaker is indicated by the presence of gas in the chamber, and deducing which if any of a number of substances a patient is allergic to by simultaneously administering doses of a group of substances and observing whether or not there is a reaction.

The most widely investigated binary group testing model is *error-free binomial* binary group testing where each unit is assumed to have an independent probability p of being defective and $q = 1 - p$ of being non-defective where p is known. This was the model proposed by Dorfman (1943) in the first formulation of the group testing problem. There have been many generalizations of this model in subsequent literature, for example unknown p, errors in observations, and units with unequal probabilities of being defective. The current work however will consider only the error-free binomial case.

The following assumptions define the binary group testing model which is to be considered here.

1. The population consists of dichotomous units called defective and non-defective.

2. Each unit is defective with probability p and non-defective with probability $q = 1-p$ where p is known.

3. All units are independent.

4. Any group of units may be tested together.

5. A test observes whether or not there are any defectives in the test group.

6. There are no errors in observations.

There is a further implicit assumption that p is small. This is not in fact a model assumption but is necessary for group testing to achieve savings in tests. In practice of course the exact value of p will not generally be known but will be an estimate based on experience, previous test history, or perhaps simply an informed guess. The likely effects of an inaccurate estimate will be investigated later, but otherwise it will be assumed that p is known exactly.

For the group testing problem defined by the above assumptions then, it is required to devise a procedure for choosing which groups are to be tested in an experiment. The aim of such a procedure, or *algorithm*, is to correctly classify the whole population as either defective or non-defective using as few tests as possible. There are three rules by which units may be classified:

1. In a test upon a single unit.

2. If a group is tested with a negative result then all units in the group are negative.

3. If every unit except one from a defective group has been classified as non-defective then the remaining unit is defective.

In fact rule 3 is a special case of the more general rule that if a subgroup of a defective group is tested negatively then the complement of the subgroup within the defective group is defective.

The simplest algorithm is to test each unit separately ("one-at-a-time testing") in which case all classifications will be made by rule 1 and the number of tests will be equal to the number of units. More efficient algorithms achieve savings in tests by classifying units using rules 2 and 3. Note, however that negative tests on units are required to classify using rules 2 and 3. It is therefore intuitively clear that greater savings in tests are achievable for fewer defective units in the population. In the worst case where all units are defective, rules 2 and 3 will never be used and the whole population will be classified by individual testing. In this case therefore, the number of tests is at least equal to the number of units in the population. In fact the number of tests would be equal to the number of units plus the number of tests performed on groups containing more than one unit. This has two implications. Firstly, the optimal non-sequential algorithm which can always solve the problem is that which tests all units individually. Therefore only sequential algorithms are of interest for this model. Secondly, no sequential algorithm can be guaranteed to classify the whole population using fewer tests than one-at-a-time testing. The expected savings in tests that group testing will be seen to achieve must therefore be weighed against the risk of using more tests than would otherwise be needed. The probability of this is one of the criteria by which algorithms will be compared in Chapter 6.

Neither of these implications hold for additive group testing as will be seen later.

Before discussing binary group testing algorithms, it will be useful to examine the Bayesian updating of the probability distribution of defective and non-defective units within the population. For some algorithms such as n-stage group testing, only the initial grouping is determined by probability beliefs and so only the prior distribution is required. However, for algorithms such as optimal entropy which select each test group to minimize the expected value of some loss function, it is necessary to update the distribution after each test.

Suppose that there are N units in the population and define the indicator functions

$$I_i = \begin{cases} 1 & \text{if the } i\text{th unit is defective} \\ 0 & \text{if the } i\text{th unit is non-defective} \end{cases} \qquad i = 1, 2, \ldots, N.$$

Then there are 2^N possible values of the series $(I_1, I_2, \ldots, I_N)$ and these correspond to the possible combinations of defective and non-defective units in the population. In the notation of search, call these combinations *target sets* and arbitrarily label them $T_1, T_2, \ldots, T_{2^N}$. Let the combination corresponding to the true classification of the population be called the *target* T, so that $T = T_j$ for some $j \in \{0, 1, \ldots, 2^N\}$. In the binomial case being considered here, the prior probability that $T = T_j$ where $T_j = (i_1, i_2, \ldots, i_N)$, $i_k \in \{0, 1\}$ for $k = 1, \ldots, N$, is

$$\text{prob}(I_1 = i_1, I_2 = i_2, \ldots, I_N = i_N) = p^{N_d} q^{N - N_d} \tag{4.1}$$

where $N_d = \sum_{k=1}^{N} i_k$ is the number of 1's in T_j.

Now consider the updating of this probability. Suppose that t tests have been performed on groups $X_1, X_2, \ldots, X_t$ and let the observed outcomes of these tests be $f(X_1, T)$, $f(X_2, T), \ldots, f(X_t, T)$. Then using Bayes theorem, the updated probability that $T = T_j$ is

$$\text{prob}(T = T_j \mid f(X_1, T), \ldots, f(X_t, T))$$

$$= \frac{\text{prob}(f(X_1, T), \ldots, f(X_t, T) \mid T = T_j)\ \text{prob}(T = T_j)}{\text{prob}(f(X_1, T), \ldots, f(X_t, T))} \qquad j = 1, \ldots, 2^N. \quad (4.2)$$

Now $\text{prob}(f(X_1, T), \ldots, f(X_t, T) \mid T = T_j)$ is the probability of having obtained the test outcomes observed up until time t given that T_j is the target, and so by Definition 2.4,

$$\text{prob}(f(X_1, T), \ldots, f(X_t, T) \mid T = T_j) = \begin{cases} 1 & \text{if } T_j \text{ is } t\text{-consistent} \\ 0 & \text{otherwise} \end{cases}.$$

Note also that

$$\text{prob}(f(X_1, T), \ldots, f(X_t, T))$$

does not depend on T_j. Therefore it is seen from (4.2) that the updated probability that $T = T_j$ is zero for inconsistent sets and is proportional to the binomial prior probability $\text{prob}(T = T_j)$ given by (4.1) for consistent sets. Hence the following two rules define the Bayesian updating process:

1. After each test, set $\text{prob}(T = T')$ equal to zero for all target sets T' which are inconsistent with the latest observation.

2. Renormalize the probabilities of all remaining consistent target sets so that

$$\sum_{j=1}^{2^N} \text{prob}(T_j) = 1.$$

Although the updating process is simple, the number of probabilities to be updated causes a problem since there are 2^N possible target sets for a population of size N. It would clearly be preferable, if possible, if the probability distribution was on the individual units rather than the combinations of them. This is easily achieved at the beginning of the experiment when all units are independent, but as the experiment proceeds and the units are tested together in various groups, the dependence between them becomes increasingly complex. However, there exists a class of algorithms which achieve great computational savings in the updating by imposing constraints on the next test set using the following result, proved by Sobel and Groll (1959) :

Lemma 4.1 Suppose that the distribution of the number of defectives in a group of units $\mathcal{G}$ is binomial, and that this group is tested with a positive result. Now suppose that a further test on a subgroup $\mathcal{G}' \subset \mathcal{G}$ also gives a positive result (so that $\mathcal{G}'$ forms a defective group). Then the updated distribution of the number of defectives in $\mathcal{G} - \mathcal{G}'$ is again binomial.

Now, a *non-mixing* algorithm is one which imposes the constraint on each choice of test group that it must be a subgroup of a defective group if one exists at that stage. Then by the above lemma, at each stage all unclassified units will belong to one of two distinct sets, a *binomial set* in which all units are independently defective with probability p, or a *defective set* which is known to contain at least one defective, where one or other of these

sets may be empty. Note that since all units within each set are identically distributed, the problem of choosing the next test group to minimize the expected value of some loss function is reduced to determining only the size of the next test group. The group is then chosen arbitrarily from the defective set if one exists, or from the binomial set otherwise. Furthermore, the prior probability p, the size of the binomial set, and the size of the defective set are sufficient to completely summarize the distribution of defectives in the population.

Non-mixing would also appear to be a natural procedure since nearly all algorithms which have been introduced belong to the non-mixing class. Moreover, Sobel and Groll (1959) showed that the loss in efficiency from imposing the non-mixing constraint is small, and sometimes zero.

In Chapter 6 non-mixing binary group testing algorithms will be investigated and compared, but first a new family of non-mixing algorithms will be introduced.

Chapter 5

Multi-Stage Stepwise Group Testing

The original group testing method of Dorfman (1943) involved testing groups of units and then testing individual units within defective groups. Sterret (1957) showed that the expected number of tests can be reduced by testing units within a defective group only until the first defective is found, and then testing the remaining units together. Analogous to the paper of Patel (1962) in which Dorfman's method was generalized to more than two stages, a multi-stage version of Sterrett's algorithm will now be developed. An expression for the expected number of tests is obtained and this is approximately minimized with respect to the number of groups in each stage by assuming p to be small. A new bifurcation technique is introduced which is seen to be a special case of multi-stage stepwise group testing.

5.1 The Stepwise Procedure

Consider a defective group of k units (as defined in Definition 4.1). Let the units be arbitrarily numbered $1, 2, \ldots, k$. Then the *stepwise procedure* for classifying all members of the group involves testing the units one by one until the first defective is found. The remaining unclassified members of the group are then tested together. If the test is negative then the remainder are declared non-defective and the classification is complete, otherwise the remainder form a new defective group and we begin again by looking for the first defective and so on. Note, however, that if a negative test is performed on the $(k-1)$th unit while searching for the first defective in a defective group then it can be inferred that the kth unit is defective. On the other hand, if a positive test is performed on the $(k-1)$th unit then the remainder consists only of the kth unit and only one further test is required.

The stepwise procedure is defined formally in the flow chart given in Figure 5.1. It is assumed that the group to be classified consists of k units and is known to be defective. As an example, consider a group of ten units, $A, B, \ldots, J$ say, and suppose that the group is known to be defective. Writing '0' for good and '1' for defective, suppose the units are as follows:

A	B	C	D	E	F	G	H	I	J
0	0	1	0	0	1	0	0	0	0

We begin by testing the units individually until the first defective C is found on the third test. The remainder (units $D-J$) are then tested together giving a positive result and thus form a new defective group. Testing the members of this group one by one, the first defective is found at F. The remainder $(G-J)$ are then tested together giving a negative

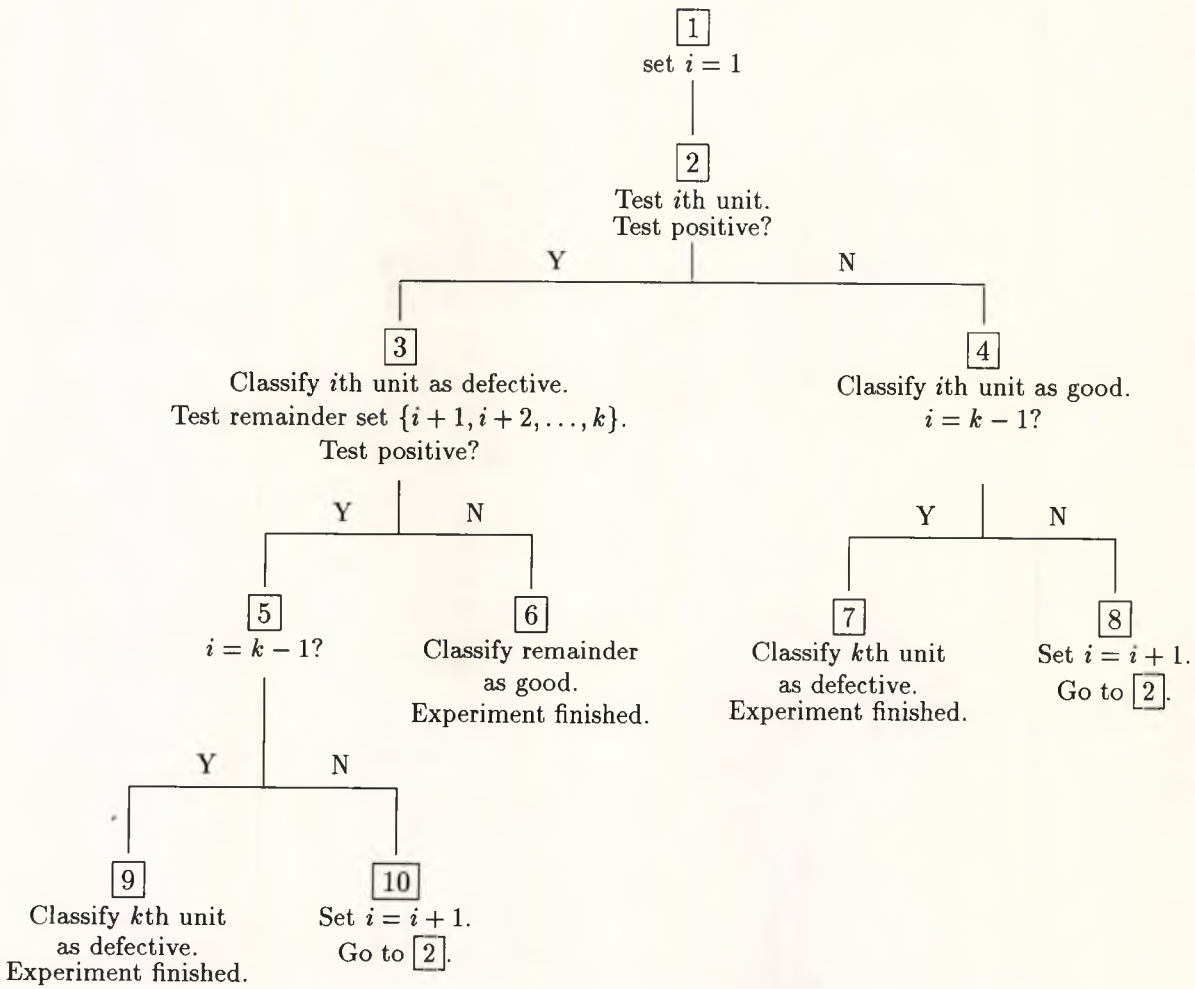

Figure 5.1: Flow chart defining the stepwise procedure for classifying a population of size k

result. Thus the classification is complete, requiring eight tests in all.

The following result gives the expected number of tests required by the stepwise procedure.

Lemma 5.1

Consider a group of k units, each of which is independently good with probability r, and suppose that the group is tested giving a positive result, i.e. the group is found to be defective. Let $T(k,r)$ be the (conditional) expected number of tests then required to classify all units in the group using the stepwise procedure. Then

$$T(k,r) = \frac{1}{1-r^k}\left((k-1)(2-r) - \frac{r^2\left(1-r^{k-1}\right)}{1-r}\right). \tag{5.1}$$

Proof

Define indicator functions

$$I_i = \begin{cases} 1 & \text{if } i\text{th unit is defective} \\ 0 & \text{if } i\text{th unit is good} \end{cases} \qquad i = 1,2,\ldots,k.$$

The tests required can be split into two types, *first-defective tests* where units are tested in order to find the first defective from a defective group (i.e. the tests performed in node $\boxed{2}$ of Figure 5.1), and *remainder tests* where, having found the first defective, the remaining units are tested together (i.e. the tests performed in node $\boxed{3}$ of Figure 5.1). Therefore,

$$\mathrm{E}(\textit{number tests}) = \mathrm{E}(\textit{number first-defective tests}) + \mathrm{E}(\textit{number remainder tests}).$$

Consider first-defective tests. It can be seen from Figure 5.1 that for $i = 1,2,\ldots,k-1$,

the ith unit is tested in node $\boxed{2}$ (i.e. in a first-defective test) if and only if the first $i-1$ units do not contain all the defectives in the group, i.e. if and only if $\sum_{j=i}^{k} I_j \neq 0$. It can also be seen that a first defective test is never performed on the kth unit, since it is either classified by a remainder test in node $\boxed{3}$, or by deduction in node $\boxed{7}$. Therefore,

$$\begin{aligned}
\mathrm{E}(\textit{no. first-defective tests}) &= \sum_{i=1}^{k-1} \mathrm{prob}(\textit{first-defective test performed on ith unit}) \\
&= \sum_{i=1}^{k-1} \mathrm{prob}\left(\sum_{j=i}^{k} I_j \neq 0 \;\middle|\; \sum_{j=1}^{k} I_j \neq 0\right) \\
&= \sum_{i=1}^{k-1} \frac{\mathrm{prob}\left(\sum_{j=1}^{k} I_j \neq 0 \;\middle|\; \sum_{j=i}^{k} I_j \neq 0\right) \mathrm{prob}\left(\sum_{j=i}^{k} I_j \neq 0\right)}{\mathrm{prob}\left(\sum_{j=1}^{k} I_j \neq 0\right)} \\
&= \sum_{i=1}^{k-1} \frac{1 \times \left(1 - r^{k-i+1}\right)}{1 - r^k} \\
&= \frac{1}{1-r^k}\left(k - 1 - \frac{r^2\left(1 - r^{k-1}\right)}{1-r}\right). \qquad (5.2)
\end{aligned}$$

Consider now the remainder tests of node $\boxed{3}$. Since a remainder test is performed each time a defective is found amongst the first $k-1$ units in the group (i.e. each time a positive test is performed in node $\boxed{2}$ of Figure 5.1), the number of remainder tests is

equal to the number of defectives amongst the first $k-1$ units. Therefore,

$$\begin{aligned}
\mathrm{E}(\textit{number remainder tests}) &= \mathrm{E}\left(\sum_{i=1}^{k-1} I_i \;\middle|\; \sum_{j=1}^{k} I_j \neq 0\right) \\
&= \sum_{i=1}^{k-1} \mathrm{prob}\left(I_i = 1 \;\middle|\; \sum_{j=1}^{k} I_j \neq 0\right) \\
&= \sum_{i=1}^{k-1} \frac{\mathrm{prob}\left(\sum_{j=1}^{k} I_j \neq 0 \mid I_i = 1\right)\mathrm{prob}\left(I_i = 1\right)}{\mathrm{prob}\left(\sum_{j=1}^{k} I_j \neq 0\right)} \\
&= \sum_{i=1}^{k-1} \frac{1 \times (1-r)}{(1-r^k)} \\
&= \frac{1}{1-r^k}(k-1)(1-r) \qquad (5.3)
\end{aligned}$$

Combining (5.2) and (5.3) the lemma is proved.

A *stepwise group testing algorithm* is defined as follows. First the population is split into groups and these groups are tested. Groups giving a negative result are eliminated from the search, their members being classified as good. Groups giving a positive test result are split into subgroups (which may be individual units) and the subgroups within each defective group are tested using the stepwise procedure. Subsequent stages continue in the same manner, classifying subgroups (or individuals) within defective groups found in the previous stage using the stepwise procedure, until the whole population has been classified. If the experiment involves s stages of grouping (including the final groups of size one), then call the method *s-stage stepwise group testing.* Thus we refer to Sterrett's original procedure as two-stage stepwise group testing.

The expected number of tests of stepwise group testing algorithms with more than two stages and with equal-sized groups in each stage will now be derived. It will be assumed that the population may be partitioned exactly into equal-sized groups as required, since this allows a general expression to be obtained for the expected number of tests which may then be minimized with respect to group sizes. This assumption will not always hold in practice of course, and groups may vary in size by one.

Define $R(s, N, \mathbf{k}, p)$ to be the expected number of tests required by an s-stage stepwise group testing algorithm to classify a population of N units, each with probability p of being defective, using a stepwise algorithm with group sizes given by $\mathbf{k} = (k_1, \ldots, k_s)$. For ease of notation write $R(s) = R(s, N, \mathbf{k}, p)$. To begin with $R(3)$ will be derived, after which $R(s)$ for general $s \geq 2$ will be obtained.

5.2 Three-Stage Stepwise Group Testing

Let the population of N units be partitioned into g_1 1st-order groups each containing k_1 members. Each of these groups is partitioned into g_2 2nd-order groups of size k_2 (so that $N = g_1 k_1 = g_1 g_2 k_2$). The experiment then consists of three stages. In the first stage all 1st-order groups are tested. All groups giving a negative test result are eliminated from the search, their members being classified as good. In the second stage we take a defective group and test the 2nd-order groups within that group using the stepwise procedure. This is repeated for all the defective 1st-order groups found in the first stage. Similarly, in the third stage the units within each defective 2nd-order group found in the second stage are tested using the stepwise procedure.

The expected number of tests, $R(3)$, required to classify all members of the population

using a 3-stage stepwise group testing algorithm is evaluated as follows. In the first stage of the experiment the 1st-order groups are tested, thus requiring g_1 tests. In the second stage the 2nd-order groups within defective 1st-order groups are classified using the stepwise procedure. Since each 2nd-order group has a prior probability q^{k_2} of containing no defectives (where, recall, $q = 1 - p$ is the probability that a unit is good), it is seen from Lemma 5.1 that the expected number of tests required for each defective 1st-order group is $T(g_2, q^{k_2})$. Since the expected number of defective 1st-order groups is $g_1(1 - q^{k_1})$, the expected number of tests required in the second stage is $g_1(1 - q^{k_1})T(g_2, q^{k_2})$. Similarly, the expected number of tests required to classify the units within each defective 2nd-order group is $T(k_2, q)$. The expected number of defective 2nd-order groups is $g_1 g_2(1 - q^{k_2})$. Therefore the expected number of tests in the third stage is $g_1 g_2(1 - q^{k_2})T(k_2, q)$. Combining the tests for each stage gives the expected number of tests as

$$R(3) = g_1 + g_1(1 - q^{k_1})T(g_2, q^{k_2}) + g_1 g_2(1 - q^{k_2})T(k_2, q).$$

5.3 s-Stage Stepwise Group Testing

The extension to s stages ($s = 2, 3, \ldots$) is straightforward. The population is partitioned into g_1 groups of size k_1. For $i = 1, 2, \ldots, s-1$, each ith-order group of size k_i is partitioned into g_{i+1} groups of size k_{i+1}, where $g_s = k_{s-1}$ and $k_s = 1$ (i.e. the highest order groups are of size one), so that $N = \prod_{i=1}^{s} g_i$. The expected number of tests of an s-stage stepwise algorithm will now be derived.

For $i = 1, \ldots, s-1$, let d_{i+1} be the random variable representing the number of defective $(i+1)$th-order groups within a single defective ith-order group. Also, for $i = 1, \ldots, s$, let

D_i be the total number of defective ith-order groups, and let r_i be the random variable representing the number of tests required to classify all the units within a single defective ith-order group. Then

$$R(s|D_1) = g_1 + D_1\mathrm{E}(r_1)$$

and so

$$R(s) = g_1 + \mathrm{E}(D_1)\mathrm{E}(r_1). \tag{5.4}$$

Also, recalling the definition of $T(k,r)$ from Lemma 5.1, for $i = 1,\ldots,s-1$,

$$\mathrm{E}(r_i|d_{i+1}) = T(g_{i+1}, q^{k_{i+1}}|d_{i+1}) + d_{i+1}\mathrm{E}(r_{i+1})$$

and so

$$\mathrm{E}(r_i) = T(g_{i+1}, q^{k_{i+1}}) + \mathrm{E}(d_{i+1})\mathrm{E}(r_{i+1}). \tag{5.5}$$

Clearly, a defective group of size one requires no further testing, and therefore

$$\mathrm{E}(r_s) = 0. \tag{5.6}$$

Therefore, from (5.4), (5.5) and (5.6), and using linearity of expectation,

$$R(s) = g_1 + \mathrm{E}(D_1)\left(T(g_2, q^{k_2}) + \sum_{i=1}^{s-1}\left(\prod_{j=1}^{i}\mathrm{E}(d_{j+1})\right)T(g_{i+2}, q^{k_{i+2}})\right). \tag{5.7}$$

Now, for $i = 2,\ldots,s-1$, consider $\mathrm{E}(D_i)$, the expected number of ith-order groups. Conditioning on D_{i-1} gives

$$\mathrm{E}(D_i|D_{i-1}) = D_{i-1}\mathrm{E}(d_i),$$

and so

$$\mathrm{E}(D_i) = \mathrm{E}(D_{i-1})\mathrm{E}(d_i)$$

which gives

$$\mathrm{E}(D_i) = \mathrm{E}(D_1) \times \prod_{j=2}^{i} \mathrm{E}(d_j).$$

Therefore, from (5.7),

$$\begin{aligned} R(s) &= g_1 + \sum_{i=1}^{s-1} \mathrm{E}(D_i) T(g_{i+1}, q^{k_{i+1}}) \\ &= g_1 + \sum_{i=1}^{s-1} \frac{N}{k_i}(1 - q^{k_i}) T(g_{i+1}, q^{k_{i+1}}). \end{aligned} \tag{5.8}$$

5.4 Comparison of Stepwise Algorithms

The proposed multi-stage stepwise algorithms will now be optimized and compared with Sterrett's original procedure, referred to here as two-stage stepwise group testing (a more thorough study of the properties of these algorithms appears in the following chapter). A previously mentioned problem arises here in that the expression in (5.8) assumes that the population may be divided exactly according to the required grouping scheme. However, group sizes differ for algorithms with different numbers of stages and for different values of p. Therefore, in each case, the expected number of tests *per unit* was minimized for a single 1st-order group, the size of which was not fixed. The problem was thus to minimize $R(s)/k_1$ with respect to g_i $(i = 2, \ldots, s)$, where $k_1 = \prod_{i=2}^{s} g_i$ and $g_1 = 1$. This was done using a computer search program.

Table 5.1 gives the expected number of tests per unit, $R(s)/k_1$, required to classify

a group of size k_1 using optimum 2, 3 and 4-stage stepwise algorithms, as well as the group sizes in each stage, for various values of p. Also tabulated is the approximation $\hat{R}(s)/k_1$ to $R(s)/k_1$, which is derived in the following section. It is seen that the proposed multi-stage algorithms can achieve substantial reductions in the expected number of tests over Sterrett's original 2-stage algorithm. The best savings are achieved when p is small. As p increases stepwise algorithms with fewer stages become the most efficient. For the whole range of tabulated values ($p \leq 0.2$), the expected number of tests per unit of each algorithm is less than 1, and thus savings in testing are achieved.

	2-stage			3-stage			4-stage		
p	k	$R(2)/k_1$	$\hat{R}(2)/k_1$	k_1, k_2	$R(3)/k_1$	$\hat{R}(3)/k_1$	k_1, k_2, k_3	$R(4)/k_1$	$\hat{R}(4)/k_1$
0.001	45	0.0458	0.0462	169, 13	0.0212	0.0216	343, 49, 7	0.0165	0.0171
0.002	32	0.0655	0.0661	110, 10	0.0344	0.0353	180, 30, 5	0.0284	0.0294
0.005	21	0.1054	0.1071	56, 7	0.0654	0.0677	100, 20, 4	0.0579	0.0610
0.01	15	0.1517	0.1553	36, 6	0.1064	0.1111	64, 16, 4	0.0986	0.1056
0.02	11	0.2196	0.2273	25, 5	0.1727	0.1840	36, 9, 3	0.1665	0.1811
0.03	9	0.2731	0.2844	16, 4	0.2286	0.2425	27, 9, 3	0.2246	0.2470
0.04	8	0.3192	0.3350	16, 4	0.2784	0.3025	18, 9, 3	0.2767	0.3022
0.05	7	0.3598	0.3786	12, 3	0.3236	0.3500	12, 6, 3	0.3250	0.3500
0.06	6	0.3969	0.4167	9, 3	0.3659	0.3911	12, 6, 3	0.3680	0.4033
0.07	6	0.4317	0.4583	9, 3	0.4040	0.4378	12, 6, 3	0.4096	0.4567
0.08	5	0.4639	0.4880	9, 3	0.4410	0.4844	8, 4, 2	0.4474	0.4850
0.09	5	0.4937	0.5240	9, 3	0.4769	0.5311	8, 4, 2	0.4830	0.5300
0.1	5	0.5229	0.5600	6, 3	0.5103	0.5500	8, 4, 2	0.5177	0.5750
0.15	4	0.6478	0.7000	4, 2	0.6526	0.7000	8, 4, 2	0.6783	0.8000
0.2	3	0.7493	0.8000	4, 2	0.7676	0.8500	8, 4, 2	0.8204	1.025

Table 5.1: Optimum group sizes and the expected number of tests per unit, $R(s)/k_1$, and corresponding approximation $\hat{R}(s)/k_1$, for a single 1st-order group.

5.5 Approximation and Upperbound to $R(s)$

An upperbound and approximation to $R(s)$, the expected number of tests of an s-stage stepwise algorithm, will now be derived for small p. The optimal group sizes for this

approximation will be found by allowing the group sizes to vary continuously.

First note the following properties of the grouping:

$$\text{(i)} \quad g_1 = \frac{N}{k_1} \tag{5.9}$$

$$\text{(ii)} \quad g_{i+1} = \frac{k_i}{k_{i+1}} \qquad i = 1, 2, \ldots, s-1 \tag{5.10}$$

$$\text{(iii)} \quad \prod_{j=1}^{s} g_j = N \tag{5.11}$$

$$\text{(iv)} \quad g_{j+1} \geq 2 \qquad j = 1, 2, \ldots, s-1 \tag{5.12}$$

(properties (i)-(iii) are due to the assumption of equal-sized groups and property (iv) holds since if $g_{j+1} = 1$ then the jth order groups and the $(j+1)$th-order groups will be the same). Now consider $(1 - q^{k_i})T(g_{i+1}, q^{k_{i+1}})$, which, by (5.1),

$$= (1 - q^{k_i}) \frac{1}{1 - q^{k_{i+1} g_{i+1}}} \left((g_{i+1} - 1)(2 - q^{k_{i+1}}) - \frac{q^{2k_{i+1}}(1 - q^{(g_{i+1}-1)k_{i+1}})}{1 - q^{k_{i+1}}} \right)$$

$$= (g_{i+1} - 1)(2 - q^{k_{i+1}}) - \frac{q^{2k_{i+1}}(1 - q^{(g_{i+1}-1)k_{i+1}})}{1 - q^{k_{i+1}}} \qquad \text{(using (5.10))} \tag{5.13}$$

$$= (g_{i+1} - 1)(2 - q^{k_{i+1}}) - q^{2k_{i+1}} \left(1 + q^{k_{i+1}} + q^{2k_{i+1}} + \cdots + q^{(g_{i+1}-2)k_{i+1}} \right)$$

$$= (g_{i+1} - 1) \left(2 - (1-p)^{k_{i+1}} \right) - (1-p)^{2k_{i+1}} \left(1 + (1-p)^{k_{i+1}} + \cdots + (1-p)^{(g_{i+1}-2)k_{i+1}} \right).$$

Now, since

$$(1 - p)^x \geq 1 - xp, \tag{5.14}$$

it follows that

$$\begin{aligned}
(1-q^{k_i})T(g_{i+1},q^{k_{i+1}}) &\leq (g_{i+1}-1)(1+k_{i+1}p) \\
&- (1-2k_{i+1}p)\,(g_{i+1}-1-k_{i+1}p(1+2+\cdots+(g_{i+1}-2)) \\
&= (g_{i+1}-1)(1+k_{i+1}p) \\
&- (1-2k_{i+1}p)\left((g_{i+1}-1)-\frac{k_{i+1}p(g_{i+1}-2)(g_{i+1}-1)}{2}\right) \\
&\leq (g_{i+1}-1)(1+k_{i+1}p) \\
&- (1-2k_{i+1}p)(g_{i+1}-1)+\frac{k_{i+1}p(g_{i+1}-2)(g_{i+1}-1)}{2} \qquad (5.15) \\
&= \frac{(g_{i+1}-1)}{2}(2+2k_{i+1}p-2+4k_{i+1}p+k_{i+1}pg_{i+1}-2k_{i+1}p)
\end{aligned}$$

i.e.

$$(1-q^{k_i})T(g_{i+1},q^{k_{i+1}}) \leq \frac{k_{i+1}p}{2}(g_{i+1}-1)(4+g_{i+1}).$$

Substituting this into (5.8) gives

$$\begin{aligned}
R(s) &\leq g_1+\frac{Np}{2}\sum_{i=1}^{s-1}\frac{1}{k_i}.k_{i+1}(g_{i+1}-1)(4+g_{i+1}) \\
&= g_1+\frac{Np}{2}\sum_{i=1}^{s-1}\left(1-\frac{1}{g_{i+1}}\right)(4+g_{i+1}) \qquad \text{(using (5.10))}
\end{aligned}$$

i.e.

$$R(s) \leq g_1 + \frac{3(s-1)Np}{2} + \frac{Np}{2}\sum_{i=1}^{s-1}\left(g_{i+1} - \frac{4}{g_{i+1}}\right)$$

$$= \hat{R}(s), \quad \text{say.} \tag{5.16}$$

However, when p is small, the inequalities in (5.14) and (5.15) are approximately equalities. Therefore $\hat{R}(s)$ provides an approximation to $R(s)$. To investigate the effectiveness of this approximation, Table 5.1 includes a column, headed $\hat{R}(s)/k_1$, which gives the approximation of $R(s)/k_1$ for the given group sizes. It can be seen that $\hat{R}(s)$ provides reasonably good approximations to $R(s)$, especially when p is small.

The approximation $\hat{R}(s)$ will now be minimized with respect to g_i $(i = 1, 2, \ldots, s)$ subject to the constraints given by (5.11) and (5.12). This will be done by allowing g_i to vary continuously (in practice of course g_i must be an integer), and using differentiation. Writing $u_i = \log g_i$, i.e. $g_i = e^{u_i}$ $(i = 1, 2, \ldots, s)$, the problem is as follows:

$$\text{minimize:} \qquad F = e^{u_1} + \frac{Np}{2}\sum_{i=1}^{s-1}\left(e^{u_{i+1}} - 4e^{-u_{i+1}}\right)$$

subject to the constraints

$$\sum_{i=1}^{s} u_i = \log N, \tag{5.17}$$

$$u_{i+1} \geq \log 2 \qquad \text{for } i = 1, 2, \ldots, s-1. \tag{5.18}$$

This minimization uses the following result.

Lemma 5.2

In the range $e^{u_{i+1}} \geq 2 \quad i = 1, 2, \ldots, s-1$ (i.e. when constraint (5.18) holds), F is convex.

Proof

Note that a function is convex if and only if the Hessian of the function is non-negative definite. Now, for $i \neq j$,

$$\frac{\partial^2 F}{\partial u_i u_j} = 0 \qquad i, j = 1, \ldots, s$$

and hence the Hessian of F is a diagonal matrix with diagonal elements

$$\frac{\partial^2 F}{\partial u_1^2} = e^{u_1} = g_1 > 0$$

and

$$\frac{\partial^2 F}{\partial u_{i+1}^2} = \frac{Np}{2}\left(e^{u_{i+1}} - 4e^{-u_{i+1}}\right) \qquad i = 1, \ldots, s-1.$$

Therefore the Hessian is non-negative definite, and hence F is convex, if and only if

$$e^{u_{i+1}} - 4e^{-u_{i+1}} \geq 0 \qquad i = 1, \ldots, s-1,$$

i.e. if and only if $e^{u_{i+1}} \geq 2$ $(i = 1, 2, \ldots, s-1)$ as required.

Now a convex function may be minimized with respect to a linear constraint using Lagrangian methods. Therefore, if we minimize F with respect to (5.17) using Lagrangian methods, and find a solution such that $e^{u_{i+1}} \geq 2$ for $i = 1, 2, \ldots, s-1$, then this solution is a valid local minimum. Further, by convexity, it is the unique minimum in the range $u_{i+1} \geq \log 2 \quad (g_{i+1} \geq 2) \quad i = 1, 2, \ldots, s-1$, and therefore the unique solution which

minimizes F satisfying both constraints (5.17) and (5.18).

Now minimize F subject to constraint (5.17) using Lagrangian methods, i.e. minimize

$$L = e^{u_1} + \frac{Np}{2}\sum_{i=1}^{s-1}\left(e^{u_{i+1}} - 4e^{-u_{i+1}}\right) - \lambda\left(\sum_{i=1}^{s} u_i - \log N\right)$$

where λ is a Lagrange multiplier.

Differentiating and setting equal to zero yields:

$$\frac{\partial L}{\partial u_1} = 0 \quad \Rightarrow \quad e^{u_1} = \lambda \tag{5.19}$$

$$\frac{\partial L}{\partial u_{i+1}} = 0 \quad \Rightarrow \quad \frac{Np}{2}\left(e^{u_{i+1}} + 4e^{-u_{i+1}}\right) - \lambda = 0 \tag{5.20}$$

with the constraint

$$\prod_{i=1}^{s} e^{u_i} = N. \tag{5.21}$$

From (5.20) we obtain

$$Npe^{2u_{i+1}} - 2\lambda e^{u_{i+1}} + 4Np = 0$$

$$\Rightarrow e^{u_{i+1}} = \frac{\lambda \pm \sqrt{\lambda^2 - 4N^2p^2}}{Np}$$

$$= \frac{e^{u_1} \pm \sqrt{e^{2u_1} - 4N^2p^2}}{Np} \qquad \text{(from (5.19))} \tag{5.22}$$

Recall that we require $e^{u_{i+1}} \geq 2$ $(i = 1, 2, \ldots, s-1)$ for the optimization to be valid.

It is easy to see that this is achieved by taking the positive root of (5.22) provided that $e^{u_1} \geq 2Np$, i.e. provided that $g_1 \geq 2Np$ (this also guarantees real roots). Note that, if this is so, then $e^{u_{i+1}}$ (and hence g_{i+1}) is constant for $i = 1, 2, \ldots, s-1$, i.e.

$$g_2 = g_3 = \cdots = g_s$$

$$= \left(\frac{N}{g_1}\right)^{\frac{1}{s-1}} \qquad \text{using (5.21)}$$

(Table 5.1 shows that, for the true expected number of tests, the optimal values of g_i are indeed almost or exactly constant for $i \geq 2$, i.e. $g_i = k_{i-1}/k_i$ is approximately constant). We now derive an equation to give the optimum value of g_1 and show that, as required, $g_1 \geq 2Np$. Substituting the positive root of (5.22) into (5.21), we obtain

$$e^{u_1}\left(\frac{e^{u_1} + \sqrt{e^{2u_1} - 4N^2p^2}}{Np}\right)^{s-1} = N$$

i.e.

$$g_1\left(\frac{g_1 + \sqrt{g_1^2 - 4N^2p^2}}{Np}\right)^{s-1} = N$$

$$\Rightarrow \quad g_1^{\frac{1}{s-1}}\left(\frac{g_1 + \sqrt{g_1^2 - 4N^2p^2}}{Np}\right) = N^{\frac{1}{s-1}}$$

$$\Rightarrow \quad g_1^{\frac{1}{s-1}}\sqrt{g_1^2 - 4N^2p^2} = N^{\frac{s}{s-1}}p - g_1^{\frac{s}{s-1}}$$

$$\Rightarrow \qquad g_1^{\frac{2}{s-1}}(g_1^2 - 4N^2p^2) \qquad = N^{\frac{2s}{s-1}}p^2 - 2p(Ng_1)^{\frac{s}{s-1}} + g_1^{\frac{2s}{s-1}}$$

$$\Rightarrow \qquad -4g_1^{\frac{2}{s-1}}N^2p^2 \qquad = N^{\frac{2s}{s-1}}p^2 - 2p(Ng_1)^{\frac{s}{s-1}}.$$

It is convenient to express this equation in terms of k_1. Using (5.9) the equation becomes

$$-4\left(\frac{N}{k_1}\right)^{\frac{2}{s-1}}N^2p^2 = N^{\frac{2s}{s-1}}p^2 - 2p\left(\frac{N^2}{k_1}\right)^{\frac{s}{s-1}}$$

$$\Rightarrow k_1^{\frac{s}{s-1}} + 4k_1^{\frac{s-2}{s-1}} - \frac{2}{p} = 0.$$

Thus the solution to the above equation gives the optimum size of the 1st-order groups. It now needs to be shown that this solution ensures that constraint (5.18) holds. Recall that this is so provided that $g_1 \geq 2Np$, i.e. provided that $k_1 \leq 1/2p$.

Let

$$f(k) = k^{\frac{s}{s-1}} + 4k^{\frac{s-2}{s-1}} - \frac{2}{p}.$$

It is therefore required to show that the equation $f(k) = 0$ has a positive root not greater than $1/2p$. Consider the first derivative of $f(k)$:

$$f'(k) = \left(\frac{s}{s-1}\right)k^{\frac{1}{s-1}} + 4\left(\frac{s-2}{s-1}\right)k^{\frac{-1}{s-1}}.$$

Since s is the number of stages in the experiment, for multi-stage algorithms we will have $s \in \{3, 4, \ldots\}$, and for such a choice of s, $f'(k) > 0$ for all $k > 0$, i.e. $f(k)$ is an increasing function of k in the range $k > 0$. Furthermore, since $f(0) < 0$ and $\lim_{k\to\infty} f(k) = \infty$, the equation $f(k) = 0$ has precisely one positive root, k^* say, and this root is the optimum

value of k_1. It also follows, since $1/2p \geq 0$, that $k^* \leq 1/2p$ if and only if $f\left(\frac{1}{2p}\right) \geq 0$.

Now,

$$f\left(\frac{1}{2p}\right) = \left(\frac{1}{2p}\right)^{\frac{s}{s-1}} + 4\left(\frac{1}{2p}\right)^{\frac{s-2}{s-1}} - \frac{2}{p}$$

$$= \left(\frac{1}{2}\right)^{\frac{s}{s-1}} \left(\frac{1}{p}\right)^{\frac{s-2}{s-1}} \left(\left(\frac{1}{p}\right)^{\frac{1}{s-1}} - \left(\frac{1}{2}\right)^{\frac{-s}{s-1}}\right)^2$$

$$\geq 0 \quad \text{since} \quad p \geq 0.$$

Hence $k^* \leq 1/2p$ as required, and the optimization is valid.

However, for the case $s = 2$, $f(0) < 0$ if and only if $p < 1/2$ and so the above argument only holds for $0 \leq p < 1/2$. Therefore this optimization is valid for two-stage algorithms only for p in this range. This does not present a problem however since p is assumed small already (in fact Ungar (1960) showed that group testing should only be considered when $p < \frac{1}{2}(3 - \sqrt{5})$ since savings in the expected number of tests are not possible for p larger than this).

Summarizing, for an s-stage stepwise group testing algorithm, the expected number of tests is approximately minimized by partitioning the population into 1st-order groups of size approximately k_1, where k_1 is the unique positive solution of

$$k_1^{\frac{s}{s-1}} + 4k_1^{\frac{s-2}{s-1}} = \frac{2}{p}. \tag{5.23}$$

In practice of course, k_1 must be an integer, and so the integer closest to the minimizing

value of (5.23) would be used. Note also that, from (5.9) and (5.11), $k = \prod_{j=2}^{s} g_j$, and so, from (5.12), $k_1 \geq 2^{s-1}$. Therefore, if for some value of s and p the minimizing value of k_1 in (5.23) is less than 2^{s-1}, then the optimum value for the size of 1st-order groups should be taken to be 2^{s-1} (this defines a bifurcation algorithm which will shortly be discussed). Table 5.2 shows the approximation to the size of 1st-order groups for various values of s and p.

	s						
p	2	3	4	5	6	7	8
0.001	45	156	280	376	439	473	491
0.002	32	97	162	206	232	244	249
0.005	20	52	77	91	98	100	128
0.01	14	32	43	48	50	64	128
0.02	10	19	23	25	32	64	128
0.05	6	9	10	16	32	64	128
0.1	4	5	8	16	32	64	128
0.15	3	4	8	16	32	64	128
0.2	2	4	8	16	32	64	128

Table 5.2: Approximation to the optimum size of 1st-order groups for an s-stage stepwise group testing algorithm

In all subsequent stages defective groups are partitioned equally into g_{i+1} subgroups where

$$g_{i+1} = \left(\frac{N}{g_1}\right)^{\frac{1}{s-1}} = k_1^{\frac{1}{s-1}} \qquad i = 1, 2, \ldots, s-1.$$

Unfortunately, (5.23) has no closed form solution in general and must be solved iteratively. However, for the cases $s = 2$ and $s = 4$, (5.23) reduces to quadratic equations with closed form solutions as follows.

For the case $s = 2$, (5.23) becomes

$$k_1^2 + 4 = \frac{2}{p}$$

and so the optimum size of first-order groups in a two-stage stepwise algorithm is approximately given by

$$k_1 = \sqrt{\frac{2-4p}{p}}$$

(recall that for $s = 2$ the above optimization is valid only for $p < 1/2$).

When $s = 4$, (5.23) yields

$$k_1^{\frac{4}{3}} + 4k_1^{\frac{2}{3}} = \frac{2}{p}$$

$$\Rightarrow \quad k_1 = \left(-2 + \sqrt{4 + 2/p}\right)^{\frac{3}{2}}.$$

5.6 A Bifurcation Algorithm

A special case of multi-stage stepwise algorithms occurs when $g_2 = g_3 = \ldots = g_s = 2$, i.e. a bifurcation algorithm. This algorithm differs from previously investigated bifurcation techniques in that, by varying the number of stages in the algorithm, the population may be initially partitioned into groups and tested, and bifurcation performed on those groups giving a positive result, as opposed to starting with bifurcation of the population as a whole (i.e. we need not have $g_1 = 1$). Therefore, whereas before we minimized the expected number of tests for fixed s with respect to group sizes, here it is interesting to fix $g_2 = g_3 = \cdots = g_s = 2$, and minimize with respect to s, the number of stages.

To compute the expected number of tests of an s-stage bifurcation algorithm, $R_b(s)$ say, first note the following properties of such an algorithm:

$$\text{(i)} \quad g_{i+1} = 2 \qquad i = 1, 2, \ldots, s-1$$

$$\text{(ii)} \quad g_1 = \frac{N}{2^{s-1}}$$

$$\text{(iii)} \quad k_i = 2^{s-i} \qquad i = 1, 2, \ldots, s$$

Therefore, from (5.13),

$$(1 - q^{k_i})T(g_{i+1}, q^{k_{i+1}}) = (2-1)(2 - q^{2^{s-i-1}}) - \frac{q^{2.2^{s-i-1}}(1 - q^{(2-1).2^{s-i-1}})}{1 - q^{2^{s-i-1}}} \qquad i = 1, \ldots, s-1$$

$$= 2 - q^{2^{s-i-1}} - q^{2^{s-i}}$$

Substituting this into the expression $R(s)$ for general multi-stage stepwise algorithms given in (5.8), gives

$$R_b(s) = g_1 + \sum_{i=1}^{s-1} \frac{N}{2^{s-i}}(2 - q^{2^{s-i-1}} - q^{2^{s-i}})$$

$$= \frac{N}{2^{s-1}} + \sum_{i=1}^{s-1} \frac{N}{2^{s-i}}(2 - q^{2^{s-i-1}} - q^{2^{s-i}})$$

which is the expected number of tests required to classify g_1 groups of size 2^{s-1} using an s-stage bifurcation algorithm.

Let the optimum number of stages be denoted by s_{opt}. Then s_{opt} will be the smallest integer larger than the root of $R_b(s+1) - R_b(s) = 0$. Now

$$R_b(s+1) - R_b(s) = \frac{N}{2^s} - \frac{N}{2^{s-1}} + \frac{N}{2^s}\left(2 - q^{2^{s-1}} - q^{2^s}\right)$$

$$= \frac{N}{2^s}\left(1 - q^{2^{s-1}} - q^{2^s}\right) .$$

Setting this equal to zero yields

$$1 - q^{2^{s-1}} - q^{2^s} = 0$$

$$\Rightarrow \quad q^{2^{s-1}} = \frac{-1+\sqrt{5}}{2}$$

$$\Rightarrow \quad s = 1 + \frac{1}{\log 2} \times \log\left(\frac{\log\left(\frac{\sqrt{5}-1}{2}\right)}{\log q}\right)$$

and therefore the optimum number of stages is given by

$$s_{opt} = 2 + \left[\frac{1}{\log 2} \times \log\left(\frac{\log\left(\frac{\sqrt{5}-1}{2}\right)}{\log q}\right)\right]$$

where [.] denotes the integer part of.

Table 5.3 gives the optimum number of stages s_{opt} and the corresponding expected number of tests per unit $R_b(s_{opt})/2^{s_{opt}-1}$ required to classify a single 1st-order group of size $2^{s_{opt}-1}$ for various values of p. It is seen that this bifurcation algorithm performs well over the whole range of p, competing well with the best algorithm in Table 5.1 for all tabulated values. Bifurcation thus provides a consistent version of the stepwise algorithm, with the added advantages of being easy to implement and having an exact expression for the optimal grouping scheme.

p	s_{opt}	$R_b(s_{opt})/k_1$
0.001	10	0.0149
0.002	9	0.0268
0.005	8	0.0569
0.01	7	0.0990
0.02	6	0.1688
0.03	5	0.2295
0.04	5	0.2800
0.05	5	0.3283
0.06	4	0.3733
0.07	4	0.4109
0.08	4	0.4474
0.09	4	0.4830
0.1	4	0.5177
0.15	3	0.6526
0.2	3	0.7676

Table 5.3: Expected number of tests per unit for optimal bifurcation algorithms

Chapter 6

An Investigation of Non-Mixing Binary Group Testing Algorithms

Since the subject of group testing was first introduced many new algorithms have been proposed. These algorithms are generally more efficient than Dorfman's original procedure in the sense of requiring fewer tests on average to classify the whole population. However, an experimenter may be unhappy to base a choice of algorithm solely on the expected number of tests and may wish for more detailed knowledge of an algorithm's performance. Typical questions one might ask are:

- What is the maximum number of tests that may be required?
- What if my estimate of p is inaccurate?
- How certain can I be that group testing will achieve savings in tests?
- What is the probability of requiring more than c tests?

Previous studies have only considered the expected number of tests of algorithms. The purpose of this chapter is to provide a more thorough examination of a number of non-mixing algorithms and to investigate their relative performances in terms of criteria other than the expected number of tests. The following criteria will be considered:

1. p.d.f. of the number of tests,
2. variance of the number of tests,
3. worst case number of tests,
4. sensitivity,

6.1 Non-Mixing Group Testing Algorithms

As was defined earlier, an *algorithm* for classifying all the units in a population is a set of decision rules which, depending on information obtained from previous test results, determines the next group to be tested. In this chapter *non-mixing* algorithms will be considered which choose each test group to be a subgroup of a defective group if one exists at that stage. It was seen that, by virtue of Lemma 4.1, at each stage all unclassified units will then belong to one of two distinct sets, a binomial set or a defective set. The sizes of these two sets are sufficient to completely summarize the probability beliefs at each stage of the search.

Suppose the population consists of N units, each of which has an independent prior probability $p = 1 - q$ of being defective, and let $S[n, m]$ denote a stage in the experiment in which there are n units still unclassified of which m form a defective set, the remaining $n - m$ forming a binomial set. When $m \geq 2$ we call this the *defective situation* or *D-*

situation. When $m = 0$, so that all unclassified units belong to a binomial set, we call this the *binomial situation* or *B-situation.* Note that if $m = 1$ then there is a single unit in a defective set so this can be classified as defective, therefore $S[n, 1] = S[n-1, 0]$.

The next test group is chosen from the defective set if this is non-empty, and from the binomial set otherwise. A non-mixing algorithm may therefore be considered as a mapping from the current state of the search to the size of the next test group:

$$A \;:\; S[n,m] \longrightarrow \begin{cases} \{1,2,\ldots,n\} & \text{if } m = 0 \\ \{1,2,\ldots,m-1\} & \text{if } m \neq 0 \end{cases} \qquad n = 1,\ldots,N;\; m = 0,2,3,\ldots,n$$

(note that in the *D*-situation the test group will not be the whole of the defective set since this is known to contain at least one defective and the result would therefore be predictable). The search is finished when $n = 0$. Write $A(n, m)$ to denote the size of the next test group specified by the algorithm A. An algorithm may therefore be completely described by $A(n, m)$ for all possible values of n and m. This will now be done for all algorithms which are to be included in the ensuing investigation.

6.1.1 Two-Stage Pooling

Consider first the original algorithm of Dorfman (1943) which will be referred to here as *pooling.* The population is partitioned into g groups of size k ($N = gk$). Each group is tested and units within defective groups are tested individually, units within non-defective groups being classified as good. Suppose that this is carried out by classifying groups one at a time, testing the units within a defective group before testing the next group. Then

the decision rules for this algorithm are given by

$$A(n,0) = \begin{cases} k & \text{if } n \equiv 0 \pmod{k} \\ 1 & \text{otherwise} \end{cases}$$

$$A(n,m) = 1 \qquad m \geq 2\,.$$

6.1.2 Multi-Stage Pooling

Dorfman's algorithm was generalized to $s \geq 2$ stages by Patel (1962). The population is split into g_1 1st-order groups of size k_1. For $i = 1, \ldots, s-1$ each ith order group of size k_i is split into g_{i+1} groups of size k_{i+1} where $k_s = 1$ (i.e. the smallest groups are of size 1). At the $(i+1)$th stage the $(i+1)$th-order groups within defective ith-order groups found in the previous stage are tested, until all units have been classified. Proceeding as for the 2-stage scheme by classifying the units within a defective group before testing the next group, the decision rules for an s-stage algorithm assuming equal sized groups throughout are

$$A(n,0) = \begin{cases} k_1 & \text{if } n \equiv 0 \pmod{k_1} \\ k_j & \text{if } n \equiv 0 \pmod{k_j},\ n \not\equiv 0 \pmod{k_{j-1}};\ j = 2, \ldots, s \end{cases}$$

$$A(n,m) = \begin{cases} k_{j+1} & \text{if } m = k_j;\ j = 1, \ldots, s-1 \\ k_j & \text{if } m \neq k_j, m \equiv 0 \pmod{k_j},\ m \not\equiv 0 \pmod{k_{j-1}};\ j = 2, \ldots, s \end{cases}$$

$$m \geq 2.$$

6.1.3 Two-Stage Stepwise Group Testing

Another generalization of Dorfman's algorithm was stepwise group testing, due to Sterret (1957), which has already been discussed in Chapter 5. This method is the same as two-stage pooling except that units within defective groups are testing using the stepwise procedure described in Section 5.1. The decision rules for this algorithm are given by

$$A(n,0) = \begin{cases} k & \text{if } n \equiv 0 \pmod{k} \\ n \bmod k & \text{otherwise} \end{cases}$$

$$A(n,m) = 1 \qquad m \geq 2\,.$$

6.1.4 Multi-Stage Stepwise Group Testing

The stepwise algorithm was generalized to $s \geq 2$ stages in the previous chapter. The decision rules for this algorithm are

$$A(n,0) = \begin{cases} k_1 & \text{if } n \equiv 0 \pmod{k_1} \\ n \bmod k_{j-1} & \text{if } n \equiv 0 \pmod{k_j},\ n \neq 0 \pmod{k_{j-1}};\ j = 2,\ldots,s \end{cases}$$

$$A(n,m) = \begin{cases} k_{j+1} & \text{if } m = k_j;\ j = 1,\ldots,s-1 \\ k_j & \text{if } m \neq k_j, m \equiv 0 \pmod{k_j},\ m \neq 0 \pmod{k_{j-1}};\ j = 2,\ldots,s \end{cases}$$

$$m \geq 2.$$

These rules may also be used to obtain the group sizes for the bifurcation algorithm described in Section 5.6 which will be referred to here as *stepwise bifurcation* to distinguish it from the following algorithm.

6.1.5 S & G Bifurcation

Sobel and Groll (1959) introduced a number of algorithms, one of which is the bifurcation algorithm which tests the whole of the binomial set in the B-situation and [m/2] of the units from the defective set in the D-situation, i.e.

$$\begin{aligned} A(n,0) &= n \\ A(n,m) &= [m/2] \qquad m \geq 2 \end{aligned}$$

where [.] denotes the integer part. This algorithm, which will be referred to as S & G *bifurcation*, has the advantage that the choice of group sizes does not depend on p and therefore requires no prior knowledge.

6.1.6 Maximum Entropy

Another algorithm proposed in [38] involves choosing the next test group in order to maximize the amount of entropy in the test. Since entropy is maximized by choosing the test set such that the probability of achieving either outcome is as close as possible to 1/2, the decision rules are given by

$$A(n,0) = \arg\left(\min_{1\leq x\leq n}\{|q^x - 1/2|\}\right)$$

$$A(n,m) = \arg\left(\min_{1\leq x\leq m-1}\left\{\left|\frac{q^x - q^m}{1-q^m} - 1/2\right|\right\}\right) \qquad m \geq 2.$$

The group sizes can in fact be computed explicitly to within ± 1 by equating the probability of observing a positive test result to $1/2$. For the B-situation this gives

$$\begin{aligned} 1 - q^x &= \frac{1}{2} \\ \Rightarrow \quad x &= \frac{\log 1/2}{\log q} \end{aligned}$$

and in the D-situation

$$\begin{aligned} \frac{1 - q^x}{1 - q^m} &= \frac{1}{2} \\ \Rightarrow \quad x &= \frac{\log\left(1/2\left(1 + q^m\right)\right)}{\log q} . \end{aligned}$$

In either case the size of the next test group will be the integer value just less than or just greater than the relevant value of x given above.

6.1.7 Optimal Algorithm

Finally, consider the optimal non-mixing algorithm which classifies the population using the smallest possible expected number of tests. Sobel and Groll (1959) used the principle of optimality from dynamic programming to derive the following recursive equations for determining the size of the test groups at each stage.

$$A(1,0) = 1$$

$$A(n,0) = \arg\left(\min_{1 \le x \le n} \{q^x A(n-x,0) + (1-q^x)A(n,x)\}\right)$$

$$A(n,m) = \arg\left(\min_{1\le x\le m-1}\left\{\left(\frac{q^x-q^m}{1-q^m}\right)A(n-x,m-x)+\left(\frac{1-q^x}{1-q^m}\right)A(n,x)\right\}\right)$$

$m \ge 2.$

An alternative method for determining the sizes of groups for the optimal algorithm was given by Hwang (1976) who observed that a non-mixing algorithm is equivalent to repeatedly searching for the first defective in a population of units which, as was seen in Section 3.3, may be considered as an alphabetic tree search. The optimal solution for alphabetic search trees, solved by Hu and Tucker (1971), is given by a "bottom-up" construction method and requires less computing time than the above recursive equations.

6.2 Expected Number of Tests

The expected number of tests has been previously investigated for all of the algorithms being studied here. However, to complement the following investigation of these algorithms, Table 6.1 gives the expected number of tests for various values of N and p.

Let $R(n,m)$ be the expected number of tests remaining when the current state of the experiment is $S[n,m]$. Then the recursion formulae given in [38] for computing the expected number of tests for several proposed algorithms are easily generalized for any non-mixing algorithm. For an algorithm given by $A(n,m)$, $n = 1,\ldots,N$; $m = 0,2,\ldots,n$, these formulae are

$$\begin{aligned}
R(1,0) &= 1\\
R(n,0) &= 1+q^{A(n,0)}R(n-A(n,0),0)+\left(1-q^{A(n,0)}\right)R(n,A(n,0))\\
R(n,m) &= 1+\frac{q^{A(n,m)}-q^m}{1-q^m}R(n-A(n,m),m-A(n,m))+\frac{1-q^{A(n,m)}}{1-q^m}R(n,A(n,m))
\end{aligned}$$

$$n = 1, \ldots, N; \quad m = 2, \ldots, n.$$

These formulae were used to obtain the entries in Table 6.1. For s-stage pooling algorithms ($s = 2, 3, 4$) the group sizes were taken to be

$$k_i = \frac{1}{p^{\frac{s-i}{s}}} \qquad i = 1, \ldots, s$$

which were shown by Patel (1962) to be approximately optimal. For the stepwise algorithms the optimal group sizes given in Table 5.1 were used. Where these grouping schemes were not exactly achievable, group sizes were taken to be as close as possible to the optimal in such a way that the sizes of groups in any stage differ by at most one, and the relevant expressions for $A(n, m)$ were adapted accordingly.

	$N = 50$			$N = 100$		
Algorithm	$p = 0.01$	$p = 0.05$	$p = 0.1$	$p = 0.01$	$p = 0.05$	$p = 0.1$
2-stage pooling	9.735	20.904	28.794	19.470	41.807	57.568
3-stage pooling	6.496	17.710	26.382	12.992	35.059	52.848
4-stage pooling	5.748	16.687	25.949	11.495	33.373	51.897
2-stage stepwise	7.637	18.015	26.144	15.181	36.031	52.288
3-stage stepwise	5.447	16.192	25.487	10.588	32.385	50.973
4-stage stepwise	4.927	16.199	25.807	9.853	32.398	51.613
Stepwise bifurcation	4.932	16.335	25.807	9.893	32.670	51.613
S & G bifurcation	4.263	16.190	29.920	8.402	35.164	66.707
Entropy	4.253	14.676	23.835	8.428	29.112	47.459
Optimal	4.243	14.555	23.750	8.320	28.959	47.375

Table 6.1: Expected number of tests of algorithms

The general conclusions to be drawn from Table 6.1 are that, in the tabulated range, stepwise algorithms outperform pooling, and for both cases more stages are better. The stepwise bifurcation algorithm, as was seen earlier, performs almost as well as the best of the other stepwise algorithms. S & G bifurcation performs well when p is small, but for

larger p performs worst of all. It is particularly interesting that entropy performs very nearly as well as the optimal algorithm throughout. Since the test group sizes can be calculated explicitly for the entropy case, entropy provides an excellent alternative to the optimal procedure which is computationally much more expensive to implement.

6.3 Probability Distribution Function

The number of tests required to classify a population of size N using a particular algorithm can be summarized by the probability distribution function (p.d.f.) of the number of tests. Let $X(n,m)$ be the number of tests remaining when the current state of the experiment is $S[n,m]$. Then the p.d.f. is given by

$$\text{prob}\left(X(N,0)=x\right) \qquad x=0,1,\ldots, \tag{6.1}$$

(in fact $X(n,m)$ is fixed and determined by which of the units are defective and which non-defective; the probabilities in the p.d.f. are with respect to the prior probability distribution on the population which in this case is binomial).

For some algorithms expressions are obtainable for the p.d.f., but except in the very simplest cases these are complicated and their derivations lengthy. Therefore recursive formulae will be obtained which give the p.d.f. for any non-mixing algorithm (for many algorithms these are in fact easier to compute than the expressions themselves).

Theorem 6.1

The p.d.f. of the number of tests required to classify a population using the non-mixing

algorithm $A(n,m)$, $n = 0,\ldots,N$; $m = 0,2,\ldots,n$ is given by the recursive equations

$$\text{prob}\left(X(0,0) = x\right) = \begin{cases} 1 & \text{for } x = 0 \\ 0 & \text{for } x \neq 0 \end{cases}$$

$$\begin{aligned} \text{prob}\left(X(n,0) = x\right) &= q^{A(n,0)} \times \text{prob}\left(X(n - A(n,0),0) = x - 1\right) \\ &+ \left(1 - q^{A(n,0)}\right) \times \text{prob}\left(X(n, A(n,0)) = x - 1\right) \end{aligned}$$

$$\begin{aligned} \text{prob}\left(X(n,m) = x\right) &= \frac{q^{A(n,m)} - q^m}{1 - q^m} \times \text{prob}\left(X(n - A(n,m), m - A(n,m)) = x - 1\right) \\ &+ \frac{1 - q^{A(n,m)}}{1 - q^m} \times \text{prob}\left(X(n, A(n,m)) = x - 1\right) \end{aligned}$$

$$n = 1,\ldots,N; \quad m = 2,\ldots,n.$$

Proof

Suppose there are n units still unclassified. Consider first the B-situation, $S[n,0]$. In this case the next test group will be of size $A(n,0)$. If this test is negative then the state of the search will be $S[n - A(n,0),0]$, whereas if the test is positive it will be $S[n, A(n,0)]$. In either case one extra test will have been performed, and so conditioning on the outcome of this test gives

$$\begin{aligned} \text{prob}\left(X(n,0) = x\right) &= \text{prob}\left(\text{next test negative}\right) \times \text{prob}\left(\, X(n,0) = x \,|\, \text{next test negative}\right) \\ &+ \text{prob}\left(\text{next test positive}\right) \times \text{prob}\left(\, X(n,0) = x \,|\, \text{next test positive}\right) \end{aligned}$$

$$= q^{A(n,0)} \times \text{prob}\left(X(n - A(n,0),0) = x - 1\right)$$

$$+ \quad \left(1 - q^{A(n,0)}\right) \times \text{prob}\left(X(n, A(n,0)) = x - 1\right) . \tag{6.2}$$

Consider now the D-situation $S[n, m]$, $m \geq 2$. The next test group will then be of size $A(n, m)$. If this test is negative then the state of the search will be $S[n - A(n,m), m - A(n,m)]$, whereas a positive test will give $S[n, A(n,m)]$. Conditioning on the outcome of this test gives

$$\begin{aligned} \text{prob}\left(X(n,m) = x\right) &= \text{prob}\left(\text{next test negative}\right) \times \text{prob}\left(X(n,m) = x | \text{ next test negative}\right) \\ &+ \text{prob}\left(\text{next test positive}\right) \times \text{prob}\left(X(n,m) = x | \text{ next test positive}\right) \\ &= \frac{q^{A(n,m)} - q^m}{1 - q^m} \times \text{prob}\left(X(n - A(n,m), m - A(n,m)) = x - 1\right) \\ &+ \frac{1 - q^{A(n,m)}}{1 - q^m} \times \text{prob}\left(X(n, A(n,m)) = x - 1\right) . \end{aligned} \tag{6.3}$$

Combining (6.2) and (6.3) with the obvious fact that zero tests are required to classify zero units completes the proof.

When implementing these recursion equations, computing time may be reduced by including the extra equation

$$\text{prob}\left(X(n,m) = x\right) = 0 \qquad \forall x < 0.$$

Figures 6.1 — 6.10 show the p.d.f. of the number of tests required by the algorithms being considered to classify a population of 50 units for various values of p.

Note that as p increases the p.d.f.s of pooling and stepwise algorithms appear to tend towards a normal-shaped curve. This is due to the fact that these algorithms involve

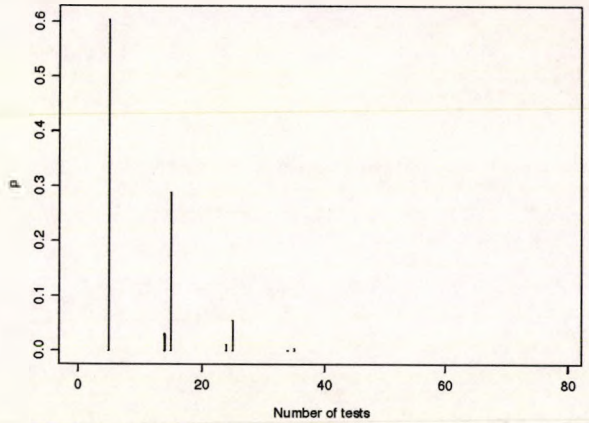

Figure 6.1 (a) Pdf of 2-stage pooling algorithm,N=50, p=0.01

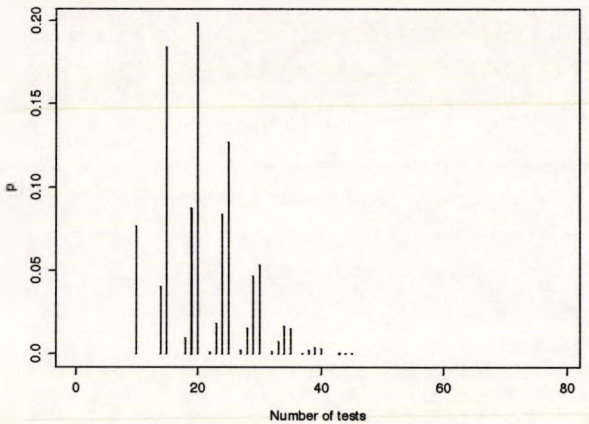

Figure 6.1 (b) Pdf of 2-stage pooling algorithm, N=50, p=0.05

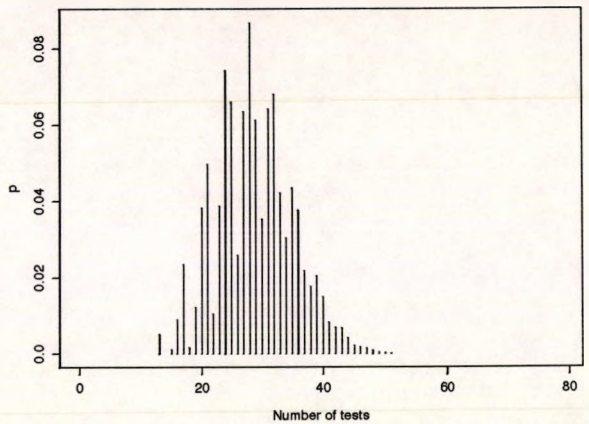

Figure 6.1 (c) Pdf of 2-stage pooling algorithm,N=50, p=0.1

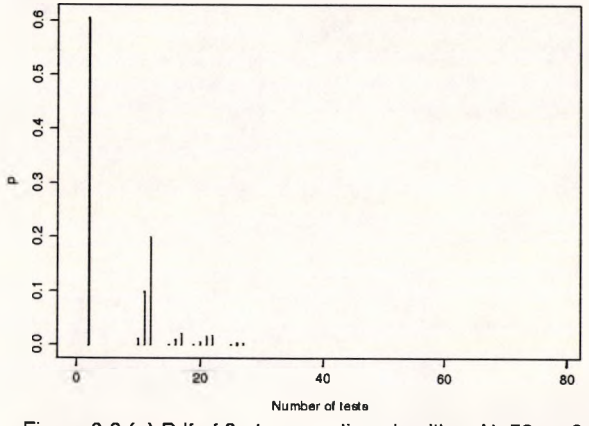

Figure 6.2 (a) Pdf of 3-stage pooling algorithm, N=50, p=0.01

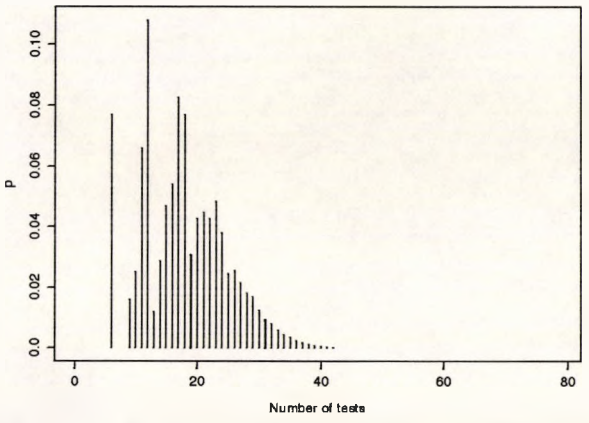

Figure 6.2 (b) Pdf of 3-stage pooling algorithm, N=50, p=0.05

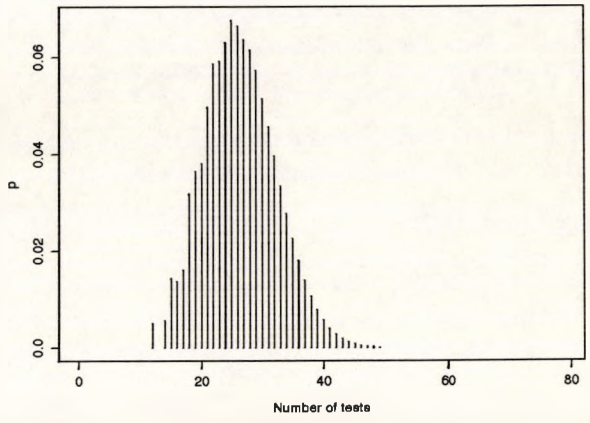

Figure 6.2 (c) Pdf of 3-stage pooling algorithm, N=50, p=0.1

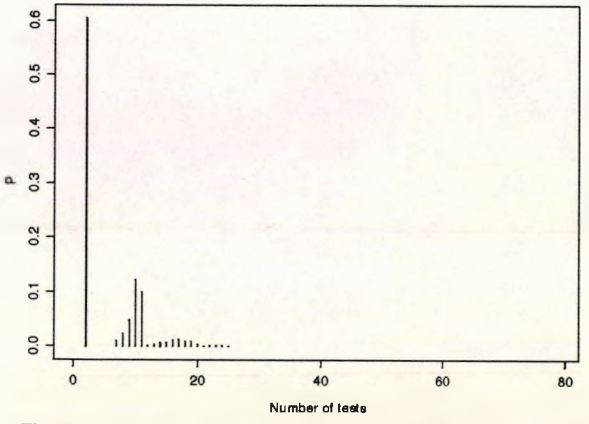

Figure 6.3 (a) Pdf of 4-stage pooling algorithm, N=50, p=0.01

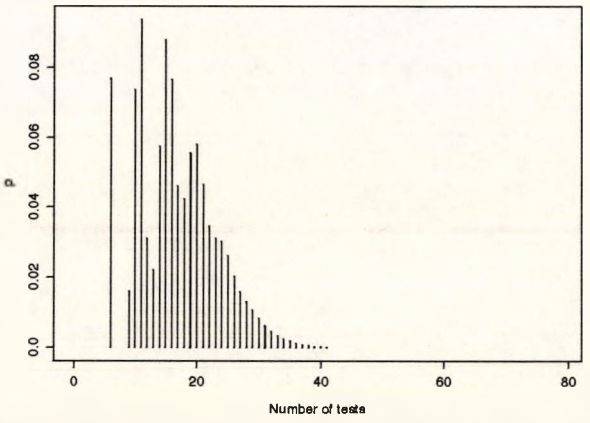

Figure 6.3 (b) Pdf of 4-stage pooling algorithm, N=50, p=0.05

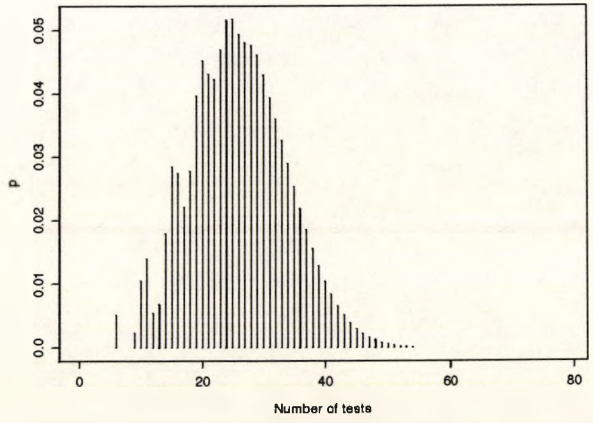

Figure 6.3 (c) Pdf of 4-stage pooling algorithm, N=50, p=0.1

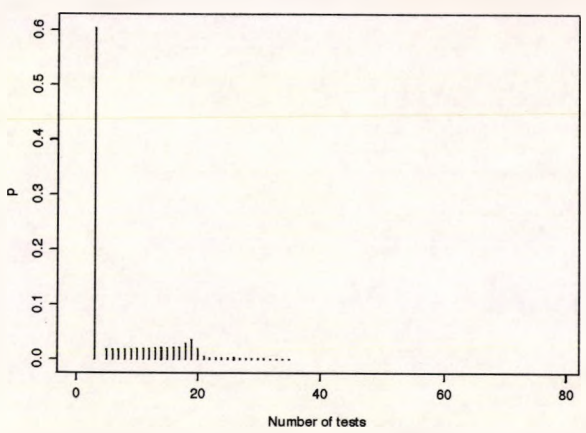

Figure 6.4 (a) Pdf of 2-stage stepwise algorithm, N=50, p=0.01

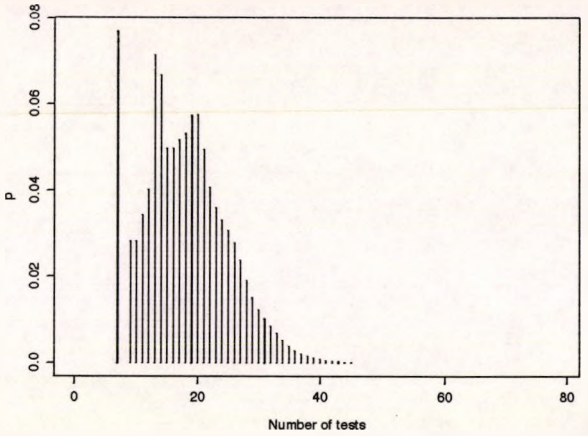

Figure 6.4 (b) Pdf of 2-stage stepwise algorithm, N=50, p=0.05

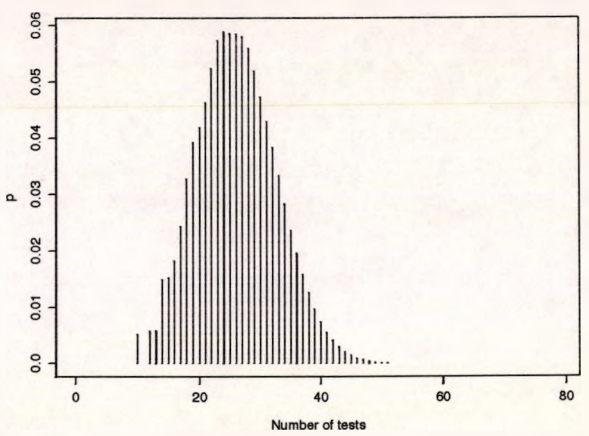

Figure 6.4 (c) Pdf of 2-stage stepwise algorithm, N=50, p=0.1

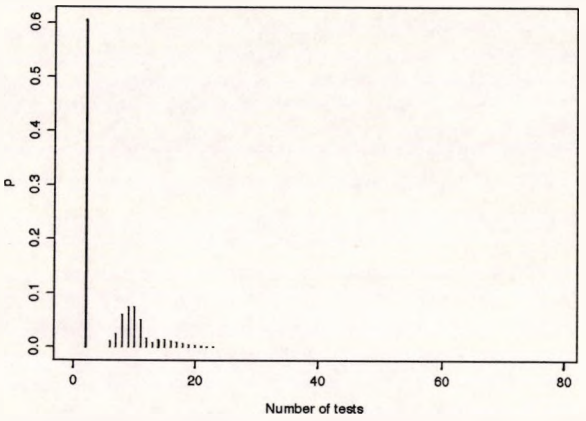

Figure 6.5 (a) Pdf of 3-stage stepwise algorithm, N=50, p=0.01

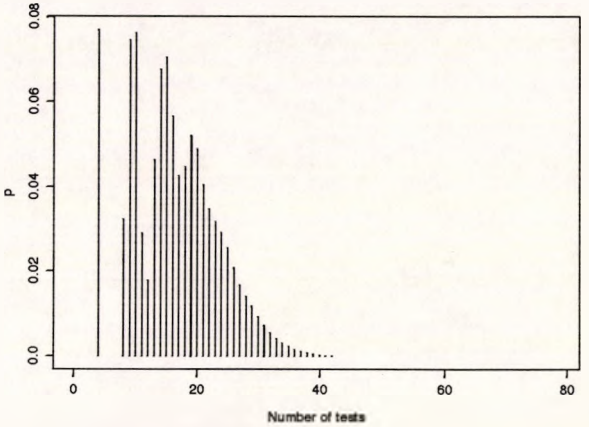

Figure 6.5 (b) Pdf of 3-stage stepwise algorithm, N=50, p=0.05

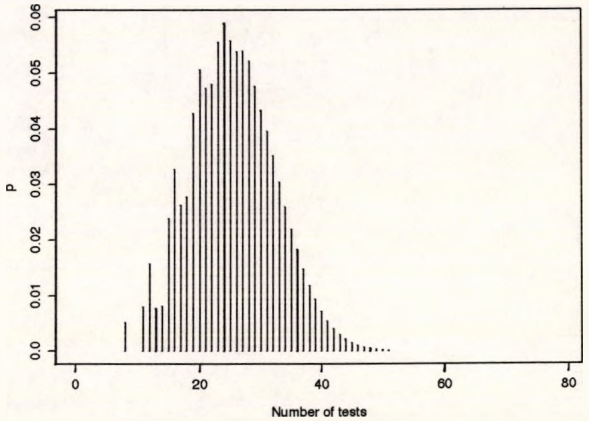

Figure 6.5 (c) Pdf of 3-stage stepwise algorithm, N=50, p=0.1

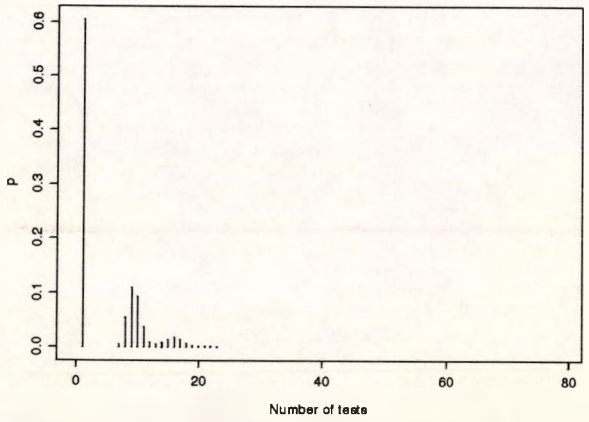

Figure 6.6 (a) Pdf of 4-stage stepwise algorithm, N=50, p=0.01

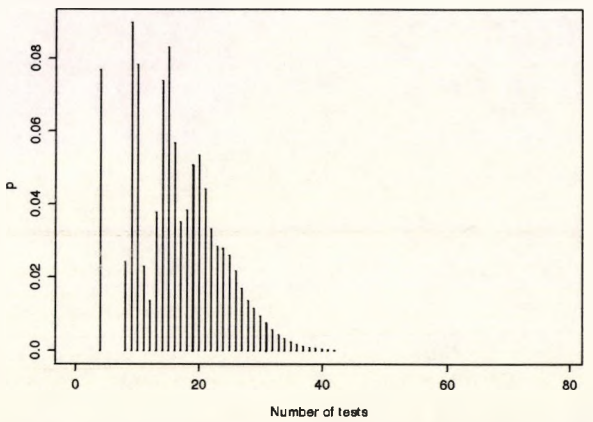

Figure 6.6 (b) Pdf of 4-stage stepwise algorithm, N=50, p=0.05

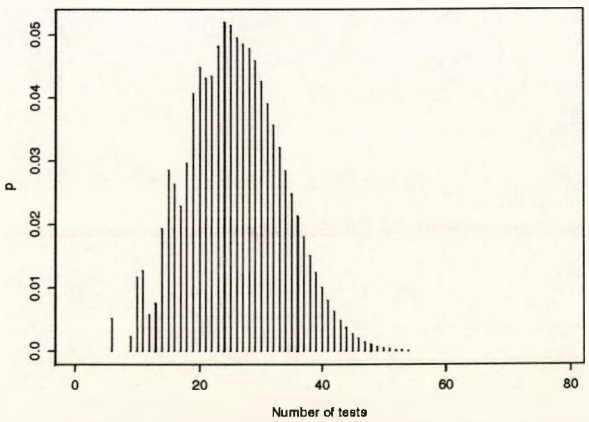

Figure 6.6 (c) Pdf of 4-stage stepwise algorithm, N=50, p=0.1

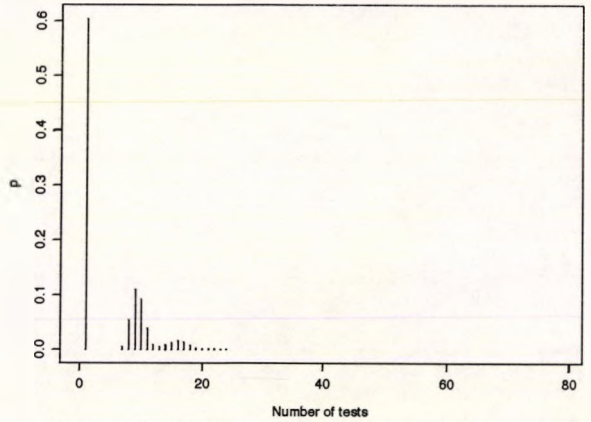

Figure 6.7 (a) Pdf of stepwise bifurcation, N=50, p=0.01

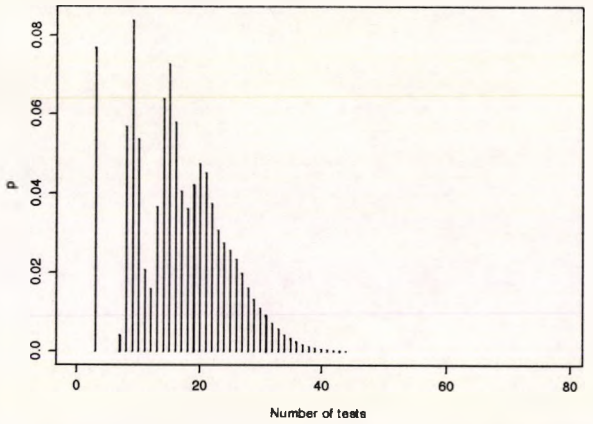

Figure 6.7 (b) Pdf of stepwise bifurcation, N=50, p=0.05

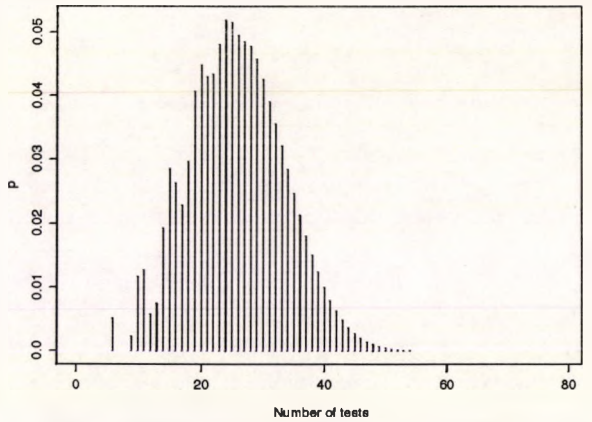

Figure 6.7 (c) Pdf of stepwise bifurcation, N=50, p=0.1

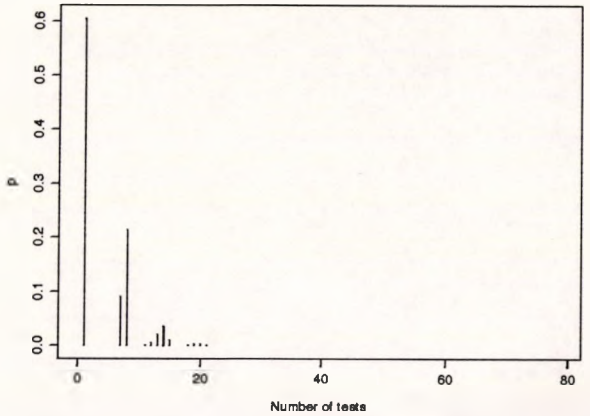

Figure 6.8 (a) Pdf of S & G bifurcation algorithm, N=50, p=0.01

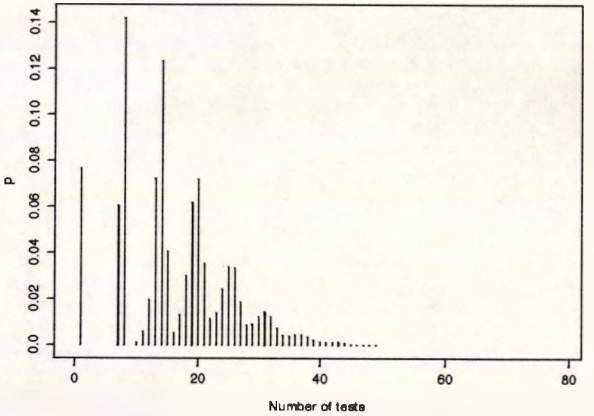

Figure 6.8 (b) Pdf of S & G bifurcation algorithm, N=50, p=0.05

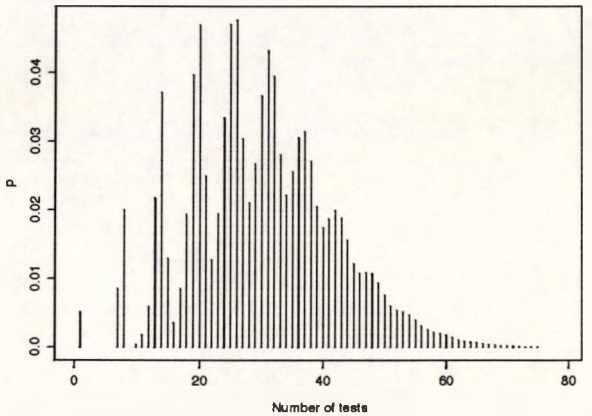

Figure 6.8 (c) Pdf of S & G bifurcation algorithm, N=50, p=0.1

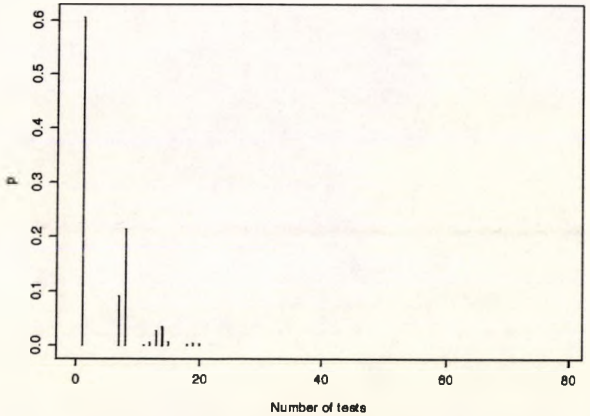

Figure 6.9 (a) Pdf of Entropy algorithm, N=50, p=0.01

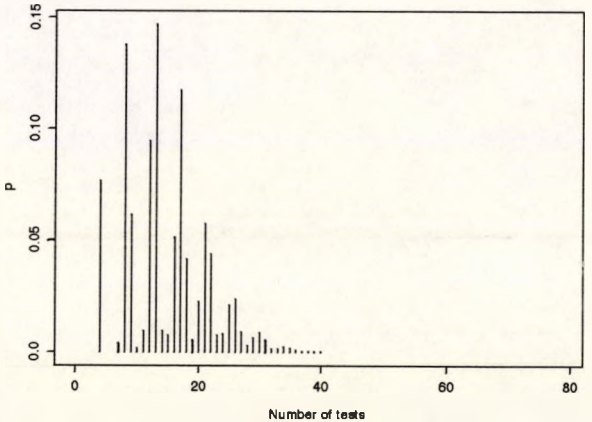

Figure 6.9 (b) Pdf of Entropy algorithm, N=50, p=0.05

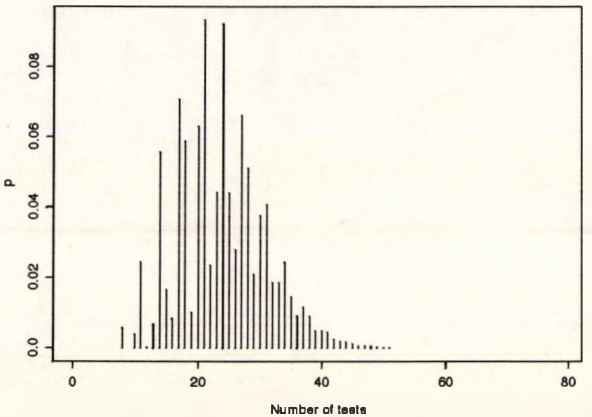

Figure 6.9 (c) Pdf of Entropy algorithm, N=50, p=0.1

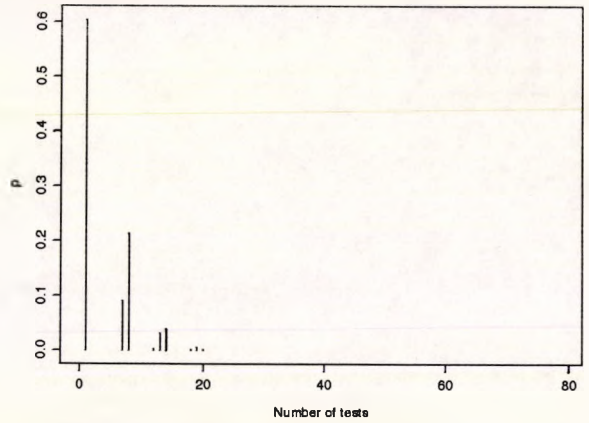

Figure 6.10 (a) Pdf of Optimal algorithm, N=50, p=0.01

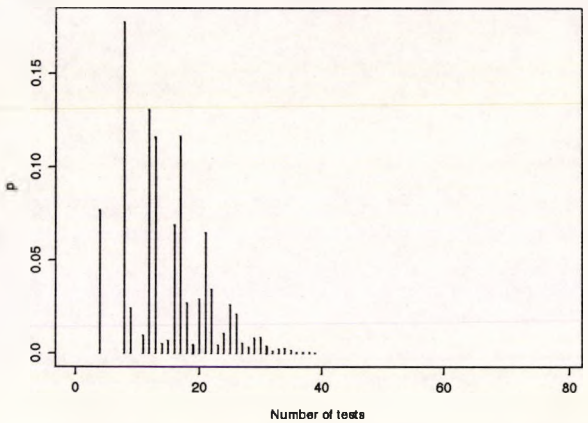

Figure 6.10 (b) Pdf of Optimal algorithm, N=50, p=0.05

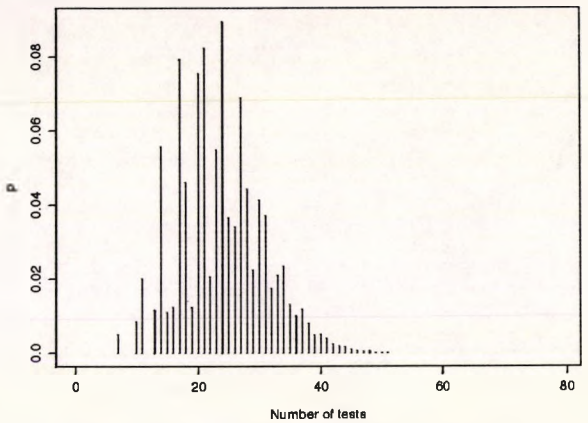

Figure 6.10 (c) Pdf of Optimal algorithm, N=50, p=0.1

partitioning the population into groups, and the total number of tests is then the sum of the number of tests in each group. Therefore, since larger p leads to smaller groups, and thus to a larger number of groups for a fixed population size, the p.d.f.s should, by virtue of the Central Limit Theorem, approach normality as p increases.

Another interesting feature of some of the figures is the occurrence of large peaks at equal intervals and other patterns such as an accompanying high value just before or just after each peak. This is again due to the grouping nature of the algorithms. Algorithms generally involve choosing groups which have a good chance of containing no defectives since this achieves savings in tests. Groups containing no defectives are classified immediately whereas groups containing defectives require further testing, the amount of which is variable. The peaks therefore correspond to the number of groups which are tested negatively.

The p.d.f. may be used to obtain the cumulative distribution function (c.d.f.). This is given by

$$\text{prob}\left(X(N,0) \leq c\right) \qquad c = 0, 1, \ldots.$$

This is of particular interest since it allows computation of the probability of requiring less than some given number of tests. Table 6.2 shows, for various values of c, the probability of requiring no more than c tests to classify 50 units using each algorithm for $p = 0.01,\ 0.05,\ 0.1$.

As well as illustrating their comparative performances, Table 6.2 also shows two important properties of the tabulated algorithms. Firstly, the table shows the probability of requiring no more than 50 tests which, since the population is of size 50, corresponds to the probability of requiring no more tests than would have been used had the units been

Algorithm	p	c							
		10	20	30	40	50	60	70	80
2-stage pooling	0.01	0.6050	0.9248	0.9925	0.9996	1.0			
	0.05	0.0769	0.5978	0.9466	0.9981	1.0			
	0.1	0.0	0.0911	0.6042	0.9656	0.995	1.0		
3-stage pooling	0.01	0.6172	0.9562	0.9993	1.0				
	0.05	0.1183	0.6674	0.9606	0.9989	1.0			
	0.1	0.0	0.1618	0.7611	0.9871	0.9999	1.0		
4-stage pooling	0.01	0.8144	0.9909	0.9999	1.0				
	0.05	0.1669	0.7385	0.9765	0.9995	1.0			
	0.1	0.0180	0.2532	0.7235	0.9655	0.9989	1.0		
2-stage stepwise	0.01	0.7187	0.9549	0.9887	0.9993	1.0			
	0.05	0.1336	0.6657	0.9543	0.9981	1.0			
	0.1	0.0052	0.2035	0.7492	0.9810	0.9998	1.0		
3-stage stepwise	0.01	0.8521	0.9922	1.0					
	0.05	0.2603	0.7371	0.9723	0.9994	1.0			
	0.1	0.0052	0.2491	0.7663	0.9809	0.9997	1.0		
4-stage stepwise	0.01	0.8697	0.9923	0.9999	1.0				
	0.05	0.2696	0.7365	0.9705	0.9992	1.0			
	0.1	0.0192	0.2577	0.7306	0.9676	0.9990	1.0		
Stepwise bifurcation	0.01	0.8698	0.9915	0.9999	1.0				
	0.05	0.2756	0.7106	0.9611	0.9985	1.0			
	0.1	0.0192	0.2577	0.7306	0.9676	0.9990	1.0		
S & G bifurcation	0.01	0.9111	0.9964	0.9999	1.0				
	0.05	0.2814	0.7280	0.9316	0.9920	0.9994	0.9999	1.0	
	0.1	0.0344	0.2325	0.5334	0.8194	0.9546	0.9921	1.0	
Entropy	0.01	0.9109	0.9975	0.9999	1.0				
	0.05	0.2826	0.7911	0.9822	0.9993	1.0			
	0.1	0.0097	0.3256	0.8277	0.9849	0.9995	1.0		
Optimal	0.01	0.9109	0.9984	1.0					
	0.05	0.2799	0.7958	0.9847	0.9994	1.0			
	0.1	0.0137	0.3380	0.8333	0.9857	0.9996	1.0		

Table 6.2: Probability of requiring no more than c tests for a population of size 50

tested individually. It is seen that, over the tabulated range, this probability is at least 0.995 for all algorithms except S & G bifurcation which has a nearly 5% chance of failing to achieve savings in tests when $p = 0.1$. Secondly, by finding the value of c for which the probability of requiring no more than c tests is almost unity (e.g. to the degree of accuracy used in the table), the maximum number of tests likely to be required by an algorithm is observed. In fact most algorithms could theoretically require many more tests than this, but the probability of this is very small.

The overall conclusions to be drawn from Table 6.2 are similar to those for the expected number of tests, namely that stepwise algorithms are generally superior to pooling, both of which perform better with more stages; S & G bifurcation performs well for very small p but badly as p increases; and entropy provides a good alternative to the optimal algorithm.

6.4 Variance

An experimenter may also wish to have knowledge of the variability of the number of tests required. Low variance of the number of tests would be advantageous since this would lead to greater consistency in the number of tests required from experiment to experiment which could be of use in the planning of an experiment. Also, it was seen in the previous section that the p.d.f.s of some algorithms tend towards normality as the number of groups increases, and so knowledge of the variance and the mean would allow predictions of properties of these algorithms by using a normal approximation. The variance of group testing algorithms was briefly considered in [38] for very small problems by considering the number of tests required for all possible outcomes, but this is clearly not plausible for larger problems since the number of combinations of defective and non-

defective units is 2^N. It would of course be possible to use the recursive equations from the previous section to obtain the p.d.f. and thus compute the variance, but it will be more efficient to derive recursive equations for computing the variance itself. This will be done by considering group testing as a binary search tree. A binary search tree is as defined in Chapter 2 except that each test can only have two possible outcomes and therefore each inner node has only two branches descending from it. A general result for the variance of binary search trees will first be obtained.

Define the *length* of a leaf to be the number of tests performed along the path from the root to the leaf. Now consider a binary tree, T say, which has n leaves with lengths $l_1, l_2, \ldots, l_n$, and let p_i be the probability that a search through T terminates at the ith leaf $(i = 1, 2, \ldots, n)$. Then the average length of a search through T, $\bar{l}$ say is

$$\bar{l} = \sum_{i=1}^{n} p_i l_i$$

and the variance of the length of a search through T, V say, is

$$V = \sum_{i=1}^{n} p_i \left(l_i - \bar{l}\right)^2 .$$

Now, let $p^{(l)}$ and $p^{(r)}$ be the respective probabilities that a search follows the left or right branch from the root of T, and let T_l and T_r be, respectively, the left and right subtrees descending from the root. Therefore, after the first test, a search is performed through T_l with probability $p^{(l)}$ and through T_r with probability $p^{(r)}$.

Suppose that T_l and T_r have n_l and n_r leaves respectively (so that $n_l + n_r = n$) and let the lengths of these leaves be $l_1^{(l)}, l_2^{(l)}, \ldots, l_{n_l}^{(l)}$ for T_l and $l_1^{(r)}, l_2^{(r)}, \ldots, l_{n_r}^{(r)}$ for T_r. Finally,

let $p_i^{(l)}$ be the probability that a search through T_l terminates at the ith leaf of T_l, and let $p_i^{(r)}$ be the probability that a search through T_r terminates at the ith leaf of T_r. Then the average length of searches through T_l and T_r, $\bar{l}^{(l)}$ and $\bar{l}^{(r)}$ say, are

$$\bar{l}^{(l)} = \sum_{i=1}^{n_l} p_i^{(l)} l_i^{(l)};$$

$$\bar{l}^{(r)} = \sum_{i=1}^{n_r} p_i^{(r)} l_i^{(r)}.$$

The variance of searches through T_l and T_r, V_l and V_r say, are

$$V_l = \sum_{i=1}^{n_l} p_i^{(l)} \left(l_i^{(l)} - \bar{l}^{(l)} \right)^2 = \sum_{i=1}^{n_l} p_i^{(l)} l_i^{(l)^2} - \bar{l}^{(l)^2};$$

$$V_r = \sum_{i=1}^{n_r} p_i^{(r)} \left(l_i^{(r)} - \bar{l}^{(r)} \right)^2 = \sum_{i=1}^{n_r} p_i^{(r)} l_i^{(r)^2} - \bar{l}^{(r)^2}.$$

We shall now establish relations between the tree T and its subtrees T_l and T_r.

The probability that a search through T terminates at the ith leaf of T_l is $p^{(l)} p_i^{(l)}$, and the probability of terminating at the ith leaf of T_r is $p^{(r)} p_i^{(r)}$. The number of tests in these cases would be $l_i^{(l)} + 1$ and $l_i^{(r)} + 1$ respectively. Therefore,

$$\begin{aligned} \bar{l} &= \sum_{i=1}^{n} p_i l_i \\ &= \sum_{i=1}^{n_l} p^{(l)} p_i^{(l)} \left(l_i^{(l)} + 1 \right) + \sum_{i=1}^{n_r} p^{(r)} p_i^{(r)} \left(l_i^{(r)} + 1 \right) \\ &= p^{(l)} \bar{l}^{(l)} + p^{(r)} \bar{l}^{(r)} + 1. \end{aligned}$$

The variance of the length of a search through T is then

$$\begin{aligned}
V &= \sum_{i=1}^{n} p_i \left(l_i - \bar{l}\right)^2 \\
&= \sum_{i=1}^{n_l} p^{(l)} p_i^{(l)} \left(l_i^{(l)} + 1 - \bar{l}\right)^2 + \sum_{i=1}^{n_r} p^{(r)} p_i^{(r)} \left(l_i^{(r)} + 1 - \bar{l}\right)^2 \\
&= \sum_{i=1}^{n_l} p^{(l)} p_i^{(l)} \left(l_i^{(l)^2} + 1 + \bar{l}^2 + 2l_i^{(l)} - 2l_i^{(l)}\bar{l} - 2\bar{l}\right) \\
&+ \sum_{i=1}^{n_r} p^{(r)} p_i^{(r)} \left(l_i^{(r)^2} + 1 + \bar{l}^2 + 2l_i^{(r)} - 2l_i^{(r)}\bar{l} - 2\bar{l}\right) \\
&= p^{(l)} \left(V_l + \bar{l}^{(l)^2} + 1 + \bar{l}^2 + 2\bar{l}^{(l)} - 2\bar{l}^{(l)}\bar{l} - 2\bar{l}\right) \\
&+ p^{(r)} \left(V_r + \bar{l}^{(r)^2} + 1 + \bar{l}^2 + 2\bar{l}^{(r)} - 2\bar{l}^{(r)}\bar{l} - 2\bar{l}\right) \\
&= p^{(l)} \left(V_l + \left(\bar{l}^{(l)} + 1 - \bar{l}\right)^2\right) + p^{(r)} \left(V_r + \left(\bar{l}^{(r)} + 1 - \bar{l}\right)^2\right). \qquad (6.4)
\end{aligned}$$

Now, a binary group testing algorithm may be represented by a binary search tree with inner nodes corresponding to the current state of the search (as given by $S[n, m]$) and the size of the next test set, and with branches corresponding to the outcomes of the tests. At each node let the left branch correspond to a negative test result, and the right branch to a positive test result. The leaves of the tree will correspond to the final classification of all units in the population. Let $V(n, m)$ be the variance of the number of tests remaining when the current state of the experiment is $S[n, m]$. By considering the tree representation of an algorithm the following result will now be proved.

Theorem 6.2

The variance of the number of tests required to classify a population using the non-

mixing algorithm $A(n,m)$, $n = 0,\ldots,N$; $m = 0,2,\ldots,n$ is given by the recursive equations

$$V(1,0) = 0$$

$$\begin{aligned} V(n,0) &= q^{t_B}\left(V(n-t_B,0)+(R(n-t_B,0)+1-R(n,0))^2\right) \\ &+ \left(1-q^{t_B}\right)\left(V(n,t_B)+(R(n,t_B)+1-R(n,0))^2\right) \end{aligned} \tag{6.5}$$

$$\begin{aligned} V(n,m) &= \frac{q^{t_D}-q^m}{1-q^m}\left(V(n-t_D,m-t_D)+(R(n-t_D,m-t_D)+1-R(n,m))^2\right) \\ &+ \frac{1-q^{t_D}}{1-q^m}\left(V(n,t_D)+(R(n,t_D)+1-R(n,m))^2\right) \end{aligned} \tag{6.6}$$

where $t_B = A(n,0)$ and $t_D = A(n,m)$ are the sizes of the next test group given by the algorithm when the current state of the experiment is $S[n,0]$ and $S[n,m]$ respectively, and $R(n,m)$ is the expected number of tests as defined in Section 6.2.

Proof

Consider first the D-situation $S[n,m]$, $m \geq 2$. In this case the size of the next test group will be $t_D = A(n,m)$. With probability

$$\frac{q^{t_D}-q^m}{1-q^m}$$

the test will be negative, in which case the left hand branch is followed from the node corresponding to $S[n,m]$, leading to the subtree with root $S[n-t_D,m-t_D]$. With prob-

ability

$$\frac{1-q^{t_D}}{1-q^m}$$

the test will be positive, in which case the right hand branch is followed leading to the subtree with root $S[n, t_D]$. Using (6.4) $V(n, m)$ may therefore be expressed in terms of the variance and expected number of tests of the subtrees descending from the node corresponding to $S[n, m]$, which gives the result in (6.6).

Now consider the B-situation $S[n, 0]$. In this case the next test group will be of size $t_B = A(n, 0)$. With probability q^{t_B} the test will be negative, in which case the left hand branch is followed from the node corresponding to $S[n, 0]$, leading to the subtree with root $S[n - t_B, 0]$. With probability $1 - q^{t_B}$ the test will be positive, in which case the right hand branch is followed leading to the subtree with root $S[n, t_B]$. Using (6.4) $V(n, 0)$ may therefore be expressed in terms of the variance and expected number of tests of the subtrees descending from the node corresponding to $S[n, 0]$, which gives the result in (6.5).

Finally, if there is only a single unclassified item, then exactly one further test is required to complete the classification, and so $V(1, 0) = 0$, which completes the proof.

Table 6.3 gives the variance of the number of tests required by each algorithm to classify a population of N units for various values of N and p.

Table 6.3 shows that the optimal and entropy algorithms generally have low variance, particularly for small p. The variance of algorithms with a small number of stages seems to be fairly constant for varying p, whereas the variance of more sophisticated algorithms seems to increase with increasing p. Note that S & G bifurcation has an extremely large variance as p increases — about three times as large as the highest of the rest when $N = 100$ and $p = 0.1$.

Algorithm	$N = 50$			$N = 100$		
	$p = 0.01$	$p = 0.05$	$p = 0.1$	$p = 0.01$	$p = 0.05$	$p = 0.1$
2-stage pooling	42.46	41.00	39.24	84.91	82.00	82.38
3-stage pooling	36.66	46.65	35.12	73.32	95.92	66.79
4-stage pooling	26.35	40.35	60.32	52.70	80.70	120.65
2-stage stepwise	49.44	46.34	43.31	76.64	92.69	86.62
3-stage stepwise	22.90	49.22	48.16	54.85	98.44	96.32
4-stage stepwise	27.85	49.88	59.57	55.70	99.76	119.15
Stepwise bifurcation	28.22	57.71	59.57	56.44	115.42	119.15
S & G bifurcation	20.29	78.41	133.79	51.13	201.71	360.33
Entropy	20.04	43.79	48.31	41.22	89.05	97.94
Optimal	19.78	43.28	48.31	48.84	88.39	97.83

Table 6.3: Variance of the number of tests of algorithms

6.5 Worst Case Number of tests

An important property of any algorithm is the maximum number of tests that may be required. This was partially discussed in Section 6.3 where Table 6.2 was used to find the value of c such that the probability of requiring more than c tests is approximately zero. However, the probabilities in Table 6.2 were obtained using the assumption that each unit in the population has an independent probability p of being defective where p is known. In practice however, it is more likely that an estimate of p will be used. If this estimate overestimates p then Table 6.2 may be used to obtain an upperbound to the maximum number of tests likely to be required. If on the other hand p is underestimated then the maximum number of tests could be much greater than indicated by Table 6.2. It would therefore be useful to know the maximum number of tests which could possibly be required (the *worst case* number of tests). Now since savings in tests are achieved only when negative group tests are performed, the worst case situation occurs when all tests are positive. This may be used to obtain recursive formulae for the worst case number of tests as follows.

Theorem 6.3

Let $WC(n,m)$ be the worst case number of tests required to classify a population of n units of which m form a defective set. Then $WC(n,m)$ for the non-mixing algorithm $A(n,m)$, $n = 0,\ldots,N$; $m = 0,2,\ldots,n$ is given by the recursive equations

$$\begin{aligned} WC(0,0) &= 0 \\ WC(n,m) &= 1 + WC(n, A(n,m)) \qquad\qquad n = 1,\ldots,N;\ \ m = 0,2,\ldots,n. \end{aligned}$$

Proof

The first of the above equations is trivial since zero tests are required to classify zero units. For the case where n units are unclassified of which m form a defective set, note that the next test group will be of size $A(n,m)$ and in the worst case situation this test will be positive. Therefore, after performing this test n units will remain unclassified of which $A(n,m)$ will form a defective set, and therefore the worst case number of tests after performing the next test will be $WC(n, A(n,m))$. This leads to the second equation.

Table 6.4 shows the worst case number of tests for each algorithm for various values of N and p.

Comparing Tables 6.4 and 6.1 it is seen that algorithms which have a low expected number of tests have poor worst case properties (with the exception of S & G bifurcation which is uniformly worst in terms of the worst case number of tests). This is due to the fact that algorithms which are better in terms of the expected number of tests generally have more sophisticated grouping schemes which involve splitting defective groups into smaller and smaller groups, and as was mentioned in Chapter 4, in the worst case situation all units

Algorithm	$N = 50$			$N = 100$		
	$p = 0.01$	$p = 0.05$	$p = 0.1$	$p = 0.01$	$p = 0.05$	$p = 0.1$
2-stage pooling	55	60	63	110	120	125
3-stage pooling	62	74	86	124	144	175
4-stage pooling	76	92	92	152	184	184
2-stage stepwise	97	93	90	193	186	180
3-stage stepwise	98	96	92	197	192	184
4-stage stepwise	99	96	94	198	192	192
Stepwise bifurcation	99	97	94	199	194	188
S & G bifurcation	243	243	243	580	580	580
Entropy	241	189	146	541	389	296
Optimal	243	192	148	580	392	298

Table 6.4: Worst case number of tests of algorithms

are eventually tested individually and therefore, in this case, the fewer tests performed on groups of more than one unit the better. Therefore, if the estimate of p might not be accurate, and in particular if the number of defectives could possibly be much larger than expected, then one should be wary of using more sophisticated algorithms such as optimal or entropy, and consider instead reducing the expected number of tests by using more stages and/or stepwise algorithms.

6.6 Sensitivity Analysis

The investigations of algorithms carried out so far have been under the assumption that p is known exactly. In the likely case that p is not known, the question arises, "how robust are the algorithms to inaccurate estimation of p?". Figures 6.11—6.19 give sensitivity curves showing the expected number of tests required against the true value of p when group sizes are obtained using various inaccurate estimates of p (no curve is shown for S & G bifurcation since this algorithm doesn't use knowledge of p to obtain each group size). Selected points from these curves are shown in Table 6.5.

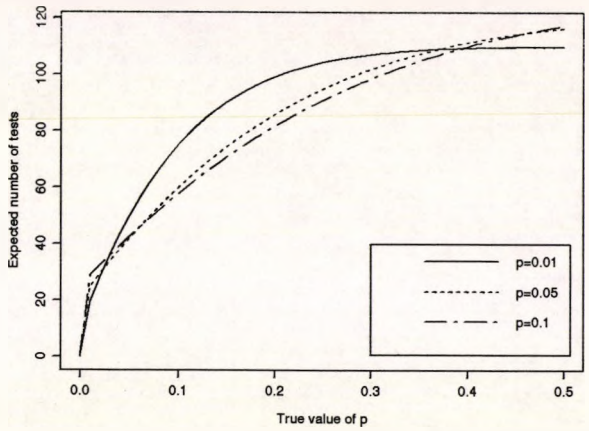

Figure 6.11 Sensitivity plot of 2-stage group testing algorithm

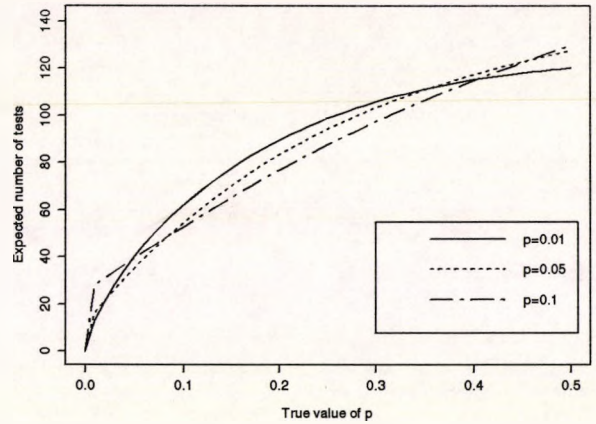

Figure 6.12 Sensitivity plot of 3-stage group testing algorithm

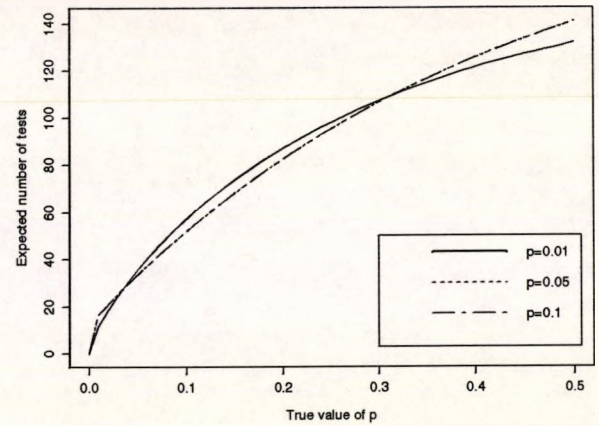

Figure 6.13 Sensitivity plot of 4-stage group testing algorithm

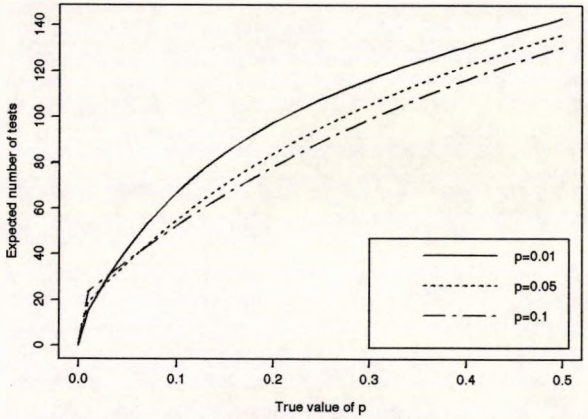

Figure 6.14 Sensitivity plot of 2-stage stepwise algorithm

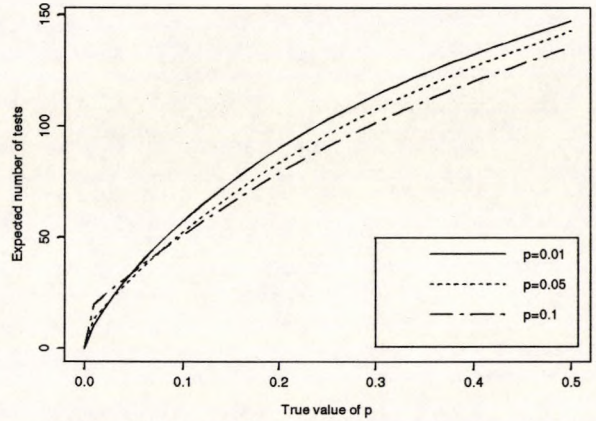

Figure 6.15 Sensitivity plot of 3-stage stepwise algorithm

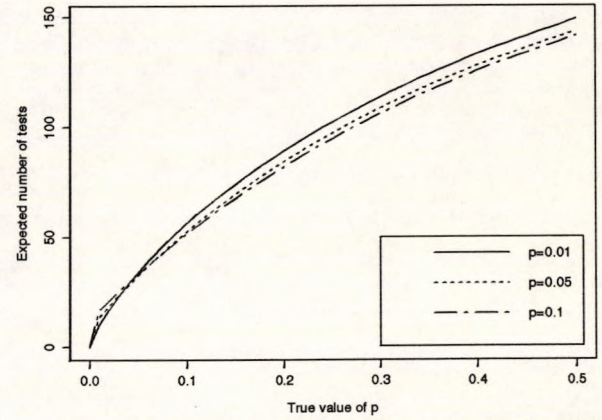

Figure 6.16 Sensitivity plot of 4-stage stepwise algorithm

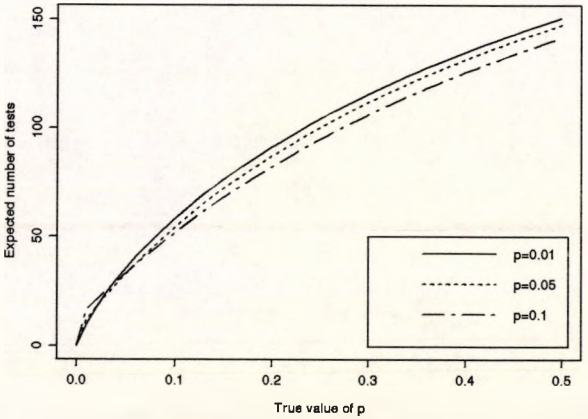

Figure 6.17 Sensitivity plot of stepwise bifurcation algorithm

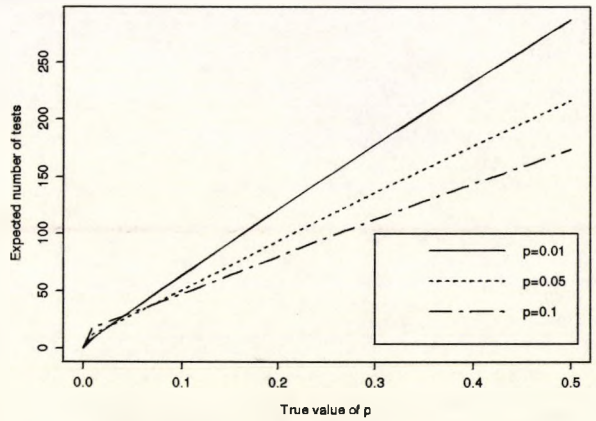

Figure 6.18 Sensitivity plot of entropy algorithm

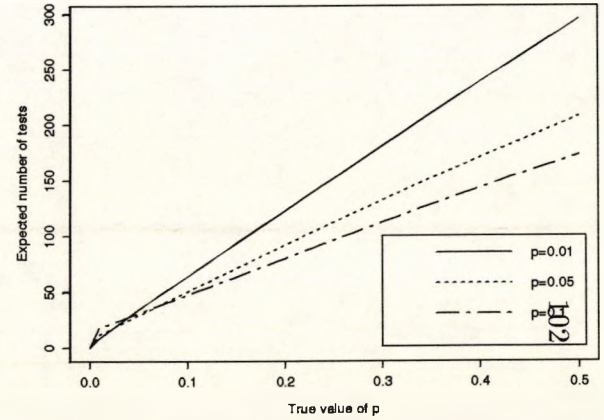

Figure 6.19 Sensitivity plot of optimal algorithm

Algorithm	Estimate of p	True value of p							
		0.01	0.05	0.1	0.15	0.2	0.3	0.4	0.5
2-stage pooling	0.01	19.47	49.81	74.74	89.97	98.99	107.05	109.36	109.89
	0.05	24.71	41.81	59.64	74.06	85.59	101.75	111.19	116.25
	0.1	28.70	42.48	57.57	70.50	81.48	98.42	109.88	117.19
3-stage pooling	0.01	12.99	39.94	62.00	77.63	89.49	105.75	115.19	120.25
	0.05	16.25	35.06	54.44	70.22	83.23	103.15	117.32	127.47
	0.1	27.98	39.45	52.85	65.26	76.76	97.25	114.76	129.69
4-stage pooling	0.01	11.50	35.53	57.21	73.67	86.86	106.88	121.23	131.73
	0.05	16.53	33.37	51.90	68.08	82.33	106.20	125.32	140.84
	0.1	16.53	33.37	51.90	68.08	82.33	106.20	125.32	140.84
2-stage stepwise	0.01	15.18	42.23	66.60	84.14	97.36	116.57	130.91	143.00
	0.05	18.72	36.03	54.60	70.35	83.81	105.51	122.35	136.10
	0.1	23.56	37.04	52.29	65.95	78.21	99.18	116.33	130.62
3-stage stepwise	0.01	10.59	34.81	57.26	75.11	90.06	113.97	132.36	147.09
	0.05	13.32	32.38	52.30	69.04	83.46	107.38	126.69	142.64
	0.1	19.87	34.49	50.97	65.71	78.92	101.55	120.12	135.53
4-stage stepwise	0.01	9.84	34.11	56.43	74.23	89.19	113.54	132.93	148.86
	0.05	13.29	32.40	52.57	69.64	84.33	108.52	127.73	143.40
	0.1	16.49	33.20	51.61	67.74	81.98	105.97	125.38	141.34
Stepwise bif.	0.01	9.89	35.34	58.00	75.96	91.03	115.46	134.73	150.40
	0.05	11.88	32.67	53.99	71.70	86.82	111.61	131.28	147.31
	0.1	16.49	33.20	51.61	67.74	81.98	105.97	125.38	141.34
Entropy	0.01	8.43	33.51	63.42	92.55	121.23	177.61	232.93	287.23
	0.05	12.04	29.11	50.66	72.16	93.52	135.59	176.57	216.31
	0.1	18.08	31.08	47.46	63.82	80.13	112.33	143.61	173.55
Optimal	0.01	8.32	34.00	64.06	93.54	122.72	180.57	238.03	295.28
	0.05	11.57	28.96	50.28	71.28	91.89	131.85	170.25	207.51
	0.1	18.01	30.98	47.38	63.73	79.99	112.06	143.22	173.09

Table 6.5: Expected number of tests to classify 100 units when p is incorrectly estimated

The table and figures show that savings in the expected number of tests are still achieved even for reasonably large inaccuracies in the estimate of p. When p is estimated to be 0.01 all algorithms require on average less tests than one-by-one testing for true values of p up to 0.15, and for true p up to 0.2 when the estimate is 0.05 or 0.1. The pooling and stepwise algorithms seem to be more robust than optimal and entropy which cease to be the best algorithms when the true value of p is 0.1, 0.15 and 0.2 for estimated

values of 0.01, 0.05 and 0.1 respectively.

As the true value of p becomes larger the simpler algorithms with fewest number of stages become the best. Therefore, if it is thought possible that the true value of p could be much larger than estimated, then pooling and stepwise algorithms should be preferred, thus reiterating the conclusion from the previous section.

Part III

ADDITIVE GROUP TESTING

Chapter 7

Additive Group Testing

The additive group testing model provides an interesting and potentially useful problem which has received very little attention in the group testing literature. For this model each unit has some effect associated with it, and the response from a group test is the sum of the effects of the units in the group. It is assumed that the population consists of *ineffective* units whose effects are constant and known (and which without loss of generality will be taken here to be zero), and *effective* units which have some larger effect. The aim of the experiment is to detect the effective factors.

Suppose the population is of size N and for $i = 1, \ldots, N$ let β_i be the effect of the ith unit. Then the following assumptions define the additive group testing model which is to be considered.

1. Each unit is effective with probability p and ineffective with probability $q = 1 - p$ where p is known.

2. All units are independent.

3. $\beta_i = 0$ if the ith unit is ineffective, $\beta_i > 0$ if the ith unit is effective.

4. Any group of units may be tested together.

5. The response from a test on a group of units is given by $\sum_{\text{Group}} \beta_i$.

6. There are no errors in observations.

There is again also the implicit assumption that p is small. Although this is not necessary to achieve savings in the expected number of tests as it was for the binary case, greater savings are made for smaller values of p.

It is seen from Assumptions 3 and 5 that the response from a group test will be zero if all units in the group are ineffective, but positive if at least one unit is effective. As with the binary case it is not known which of the units from a positively tested group are effective, nor how many effective units there are. However, greater inferences may be drawn in the additive case since if a subgroup of a previously tested group is tested then the difference in the two responses gives the result for the complement of the subgroup as well, whereas in the binary case the result for the complement may only be inferred when the tested subgroup is non-defective.

Analogous to Definition 4.1, define an *effective group* to be a group for which the sum of the effects of the units within the group is known, but for which the sum of effects of the units within any proper subset of the group is not known.

Note that in Assumption 3 nothing further is assumed about effective units other than that their effects are positive. It is necessary that effective units have effects of the same sign since otherwise effects of equal size but with opposite signs would cancel each other out if tested together in the same group (this will be discussed more fully in the chapter

on factor screening).

The model defined by the above assumptions was first considered in the group testing literature by Pfeifer and Enis (1978) who called it the *modified binomial* model. Although a binomial prior distribution is also assumed here, models with an additive response are of general interest whatever prior is assumed, and hence such models will be referred to as additive group testing, and the particular case being considered here as binomial additive group testing. An alternative additive group testing model which has been previously investigated (e.g. Sobel (1968)), but which is not considered here, is the case where the effects of effective units are constant and known. This problem is known as *hypergeometric* group testing since a group test reveals the exact number of effective units in the group.

Pfeifer and Enis (1978) investigated the properties of a 2-stage pooling algorithm. As with the binary case this algorithm involves testing groups of units and then testing individual units within effective groups. In the additive case however it is only necessary to test individual units until all the effective units have been found, since at this point the cumulative sum of individually tested units will be equal to the sum of the whole group, and so it can be inferred that the rest of the group is ineffective.

This algorithm belongs to the class of *hierarchical* models defined by Hwang, Pfeifer and Enis (1981), the distinguishing feature of which is that two units can be tested together in a group only if they have identical test histories, i.e. if each previously tested group either contained both of them or neither of them. Hwang, Pfeifer and Enis (1981) solved the optimal hierarchical algorithm by showing that the defining recursive equations of the optimal procedure may be expressed in a certain form which Glassey and Karp (1976) showed is solved by the power-of-two rule. This is a simple tree construction method

which begins by testing the whole population and then partitions all subsequent effective groups of size n into subgroups of size k and $n-k$, where k is the unique power of 2 satisfying $n/3 \leq k < (2n)/3$.

In this chapter pooling algorithms with more than two stages will be considered. Although an explicit and simple expression is available for the optimum algorithm, pooling algorithms are still of interest as they offer certain practical advantages. In particular, each unit may need to be included in approximately $1 + \log_2 N$ group tests when using the optimal algorithm compared with just s group tests for an s-stage pooling algorithm. Moreover, if there are any effective units in the population then at least one unit will be tested at least $\log_2 N$ times when using the optimal algorithm. Although this may often not be a problem, in some cases it may be undesirable to test any single unit too many times, for example blood testing or where testing is destructive. Other advantages include ease of implementation and allowing simultaneous testing of independent groups by different experimenters.

7.1 Multi-Stage Pooling Algorithms for Additive Group Testing

As for the binary case, an s-stage pooling algorithm involves partitioning the population into g_1 1st-order groups of size k_1. For $i = 1, 2, \ldots, s-1$, each ith-order group of size k_i is partitioned into g_{i+1} groups of size k_{i+1}, where $g_n = k_{n-1}$ and $k_n = 1$ (i.e. the highest order groups are of size 1). For the following analysis it will be assumed that this grouping

scheme is exactly achievable so that

$$g_{i+1}k_{i+1} = \begin{cases} N & \text{for } i = 0 \\ k_i & \text{for } i = 1, \ldots, s-1. \end{cases} \tag{7.1}$$

In practice if this is not the case then group sizes may differ in size by one.

To compute the expected number of tests of an s-stage pooling algorithm the following result will be used.

Lemma 7.1

Suppose that a group of k units, each of which is independently ineffective with probability r, is tested with a positive result so that the sum of the effects of the units is known. If the units are then tested individually until the whole group has been classified, then the expected number of tests required, $T_A(k, r)$ say, is

$$T_A(k, r) = \frac{1}{1 - r^k}\left(k - 1 - \frac{r^2(1 - r^{k-1})}{1 - r}\right). \tag{7.2}$$

Proof

Let the k units be arbitrarily ordered and define indicator functions

$$I_i = \begin{cases} 1 & \text{if } i\text{th unit is effective} \\ 0 & \text{if } i\text{th unit is ineffective} \end{cases} \qquad i = 1, 2, \ldots, k.$$

Now when testing the units individually, the cumulative sum of the effects of tested units will be equal to the sum for the whole group when all effective units in the group have been tested, but less than this if any effective units remain untested. Furthermore, the

kth unit will never need to be tested since its effect can be obtained by subtracting the sum of the effects of the first $k-1$ units from the sum for the whole group. Therefore, for $i = 1, 2, \ldots, k-1$, the ith unit is tested if and only if the first $(i-1)$ units do not contain all the effective units, i.e. if and only if

$$\sum_{j=1}^{i-1} I_j \neq \sum_{j=1}^{k} I_j$$

$$\Leftrightarrow \sum_{j=i}^{k} I_j \neq 0.$$

Therefore the expected number of tests is given by

$$T_A(k,r) = \sum_{i=1}^{k-1} \text{prob}(i\text{th unit tested})$$

$$= \sum_{j=1}^{k-1} \text{prob}\left(\sum_{j=i}^{k} I_j \neq 0 \;\middle|\; \sum_{j=1}^{k} I_j \neq 0\right) \qquad \text{(recalling that the group is effective)}$$

$$= \sum_{i=1}^{k-1} \frac{\text{prob}\left(\sum_{j=1}^{k} I_j \neq 0 \;\middle|\; \sum_{j=i}^{k} I_j \neq 0\right) \text{prob}\left(\sum_{j=i}^{k} I_j \neq 0\right)}{\text{prob}\left(\sum_{j=1}^{k} I_j \neq 0\right)}$$

(using Bayes Theorem)

$$= \sum_{i=1}^{k-1} \frac{1 \times \left(1 - r^{k-i+1}\right)}{1 - r^k}$$

$$= \frac{1}{1-r^k}\left(k - 1 - \frac{r^2\left(1 - r^{k-1}\right)}{1-r}\right)$$

thus proving the lemma.

The expected number of tests, $R_A(s)$ say, required by an s-stage pooling algorithm for the additive model can now be derived. In the first stage all 1st-order groups are tested, thus requiring g_1 tests. For $i = 1, 2, \ldots, s-1$, the $(i+1)$th stage consists of testing the $(i+1)$th-order groups within effective ith-order groups found in the previous stage, thus by definition requiring an average of $T_A(g_{i+1}, q^{k_{i+1}})$ tests per effective ith-order group. Since there are in total N/k_i ith-order groups, each with prior probability $(1-q^{k_i})$ of being effective, the expected number of tests required in the $(i+1)$th stage is

$$\frac{N}{k_i}(1-q^{k_i})T_A(g_{i+1}, q^{k_{i+1}}) \qquad i = 1, 2, \ldots, s-1.$$

Therefore the total expected number of tests is given by

$$\begin{aligned}
R_A(s) &= g_1 + \sum_{i=1}^{s-1} \frac{N}{k_i}(1-q^{k_i})T_A(g_{i+1}, q^{k_{i+1}}) \\
&= g_1 + \sum_{i=1}^{s-1} \frac{N}{k_i}(1-q^{k_i})\frac{1}{1-q^{k_{i+1}g_{i+1}}}\left(g_{i+1} - 1 - \frac{q^{2k_{i+1}}(1-q^{k_{i+1}(g_{i+1}-1)})}{1-q^{k_{i+1}}}\right) \qquad \text{(using (7.2))} \\
&= g_1 + N\sum_{i=1}^{s-1}\left(\left(\frac{1}{k_{i+1}} - \frac{1}{k_i}\right) - \frac{1}{k_i}\,\frac{q^{2k_{i+1}}(1-q^{k_{i+1}(g_{i+1}-1)})}{1-q^{k_{i+1}}}\right) \qquad \text{(using (7.1))} \\
&= g_1 + N\left(1 - \frac{1}{k_1} - \sum_{i=1}^{s-1}\frac{1}{k_i}\,\frac{q^{2k_{i+1}}(1-q^{k_{i+1}(g_{i+1}-1)})}{1-q^{k_{i+1}}}\right) \qquad \text{(since } k_s = 1\text{)} \\
&= N\left(1 - \sum_{i=1}^{s-1}\frac{1}{k_i}\,\frac{q^{2k_{i+1}}(1-q^{(k_i-k_{i+1})})}{1-q^{k_{i+1}}}\right). \qquad \text{(using (7.1))}
\end{aligned}$$

This clearly shows that pooling algorithms are always profitable in the additive case, whatever the value of p. Furthermore, the summation term within the brackets gives the average proportional saving achieved over one-at-a-time testing.

Table 7.1 shows, for various values of p, the optimal 2, 3 and 4-stage pooling algorithms and the corresponding expected number of tests per unit (the optimal group sizes were obtained using a computer search). It is seen that the proposed multi-stage algorithms require fewer tests on average then the 2-stage algorithm of Pfeifer and Enis (1978), with 4 stages being better than 3. The savings achieved are largest when p is small. For example, for $p = 0.1$ the 4-stage scheme requires approximately 16% fewer tests on average than the 2-stage scheme, for $p = 0.01$ and $p = 0.005$ the savings are approximately 46% and 54% respectively.

	2-stage		3-stage		4-stage	
p	k	$R(2)$	k_1, k_2	$R(3)$	k_1, k_2, k_3	$R(4)$
0.005	21	0.1006	56, 7	0.0568	100, 20, 4	0.0464
0.01	15	0.1424	36, 6	0.0899	64, 16, 4	0.0768
0.05	7	0.3169	16, 4	0.2537	27, 9, 3	0.2377
0.1	5	0.4429	9, 3	0.3849	12, 4, 2	0.3716
0.15	4	0.5353	9, 3	0.4868	12, 4, 2	0.4737
0.2	4	0.6096	6, 2	0.5680	8, 4, 2	0.5566
0.3	3	0.7223	4, 2	0.6950	8, 4, 2	0.6878
0.4	3	0.8080	4, 2	0.7876	8, 4, 2	0.7855
0.5	2	0.8750	4, 2	0.8594	8, 4, 2	0.8589

Table 7.1: Group sizes and expected number of tests per unit for optimum pooling algorithms when the response is additive

Pooling algorithms are compared to the optimal hierarchical algorithm in Table 7.2. The table shows the expected number of tests required to classify a population of 100 units for various values of p. The tabulated entries for the optimal algorithm were obtained using the recursive equations derived by Hwang, Pfeifer and Enis (1981). For the pooling

p	2-stage pooling	3-stage pooling	4-stage pooling	optimal
0.005	10.06	5.68	4.64	4.14
0.01	14.25	8.94	7.69	6.91
0.05	31.73	25.59	23.67	22.86
0.1	44.29	38.54	37.30	36.51
0.15	53.53	48.76	47.58	46.99
0.2	60.96	57.02	55.85	55.52
0.3	72.27	69.50	69.01	68.77
0.4	80.86	78.76	78.78	78.55
0.5	87.50	85.94	86.09	85.89

Table 7.2: Expected number of tests to classify 100 units using pooling algorithms and optimal algorithm

algorithms the optimal grouping schemes shown in Table 7.1 were used. Where these grouping schemes were not exactly achievable, group sizes were taken to be as close as possible to the optimal in such a way that the group sizes in any stage differ by at most one. It is seen that the multi-stage pooling algorithms are very efficient, with the percentage ratio of the expected number of tests of the optimal algorithm to that of the 4-stage algorithm varying from about 90% to about 99.75% over the tabulated range of p. The proposed multi-stage algorithms therefore offer efficient alternatives to the optimal algorithm with the practical advantages of pooling algorithms.

Part IV

FACTOR SCREENING

Chapter 8

Factor Screening with Unknown Error Variance

A problem which arises in certain experimental situations is where the number of explanatory variables is large. In this case the number of runs required to model the relationship between the response variable and the input variables may lead to prohibitively expensive experimental plans. For example, if there are 20 continuous input variables and it is required to estimate all main effects and two-way interactions then over 200 runs are required. This increases quadratically as the number of variables increases and is made worse when more complex models are considered, e.g. with non-linear effects and/or higher-order interactions. In the presence of test error extra runs are also required to estimate the error variance.

However, in many situations it may turn out that only a few of the explanatory variables actually have any real effect on the response. In such contexts factor screening may be used as a preliminary phase to an experiment in order to determine, or "screen out",

the important input variables, thus making subsequent experimentation more efficient and allowing the relationship between the response and the inputs to be studied in greater detail.

Suppose there are N factors each taking two levels and consider the first-order model:

$$y = \beta_0 + \sum_{j=1}^{N} x_j \beta_j + \epsilon \tag{8.1}$$

where y is the response, β_0 is a constant term, $x_j = \pm 1$ is the level of the jth factor, β_j is the effect of the jth factor, and $\epsilon \sim N(0, \sigma^2)$ is an error term. In factor screening it is not supposed that model (8.1) represents the true relationship between the response and the input factors, but only that it may be used to determine which factors are effective. Therefore, in practice, it is not necessary that (8.1) provides a good model of reality, but only that factors which are effective have a significantly non-negligible main effect when the model is fitted (this will be considered more specifically in the discussion of factor screening assumptions which follows). The model will however be assumed to be true for the development of the factor screening methods which follow.

Although a factor screening experiment considers factors at just two levels, continuous variables and factors with more than two levels may also be considered provided that the two levels used in the experiment are chosen in such a way that a change from one level to the other causes a significant change in the response for effective variables.

The idea of using group testing methods to find the effective factors in an experiment was first discussed by Watson (1961) (hereafter this paper will be referred to as WAT). By simultaneously varying the levels of groups of factors and observing whether or not this significantly affects the response, inferences may be drawn about the individual factors.

Consider a two-stage group testing plan in which the population is partitioned into g groups of size k. For the purpose of the ensuing work it will be assumed that (8.1) provides an accurate model of reality and that the following assumptions hold. These assumptions are those given in WAT and have become generally standard in subsequent factor screening literature. Although the model defined by these assumptions is clearly impractical, it will be argued shortly that in practice factor screening may still be applied even if some assumptions do not hold.

(i) all factors have, independently, the same known prior probability p of being effective (with $q = 1 - p$),

(ii) effective factors have the same effect, $\Delta > 0$,

(iii) there are no interactions present,

(iv) the required designs exist,

(v) the directions of effects of effective factors are known,

(vi) the errors of observations are independent normal with known variance σ^2,

(vii) $N = gk$ where g is the number of groups and k is the number of factors per group.

There is also the implicit assumption that p is small which allows screening to be performed using fewer runs than designs which consider all factors individually.

The two-stage factor screening method proposed in WAT is as follows. Each factor is considered at two levels, $x_j = \pm 1$. By Assumption (v) we may define the upper level of a factor to be that which produces the largest positive response. The factors are divided into equal sized groups (by Assumption (vii) this division is exact). Each group is then treated as an individual factor, called a *group-factor*, taking two levels, the upper level when all

factors in the group are at their upper level, and the lower level when all factors in the group are at their lower level. A group-factor is said to be effective if varying the group-factor from one level to the other produces a non-zero change in the expected value of the response. Therefore, by assumptions (iii) and (v), a group-factor is effective if at least one factor in the group is effective, and clearly the converse is also true. The group-factors are run in an experiment and Assumption (iv) states that the required designs exist. The group-factors are then tested for significance and factors from significant groups are tested in a second stage. Those found to be significant are classified as effective.

In fact Assumptions (i)-(vii) are not as limiting as they appear but are mainly made in order to help the development and analyses of factor screening methods, as will now be discussed.

Assumption (i) is required to obtain the optimum group size. In practice the exact value of p may not be known and an estimate of the likely proportion of effectives would be used. The sensitivity analyses in Section 6.6 showed that group screening can still be profitable even if this initial estimate is inaccurate. Alternatively, methods for unknown $p$may be used (e.g. Sobel and Groll (1966)). If it is not reasonable to assume constant p for all factors then different grouping schemes may be used, see for example Ottieno and Patel (1984).

Assumption (ii) is not in fact required but acts as an example of the size of effective factors which allows an analysis of the properties of factor screening methods, as will be seen in the following sections. In practice it is not necessary that the effects of effective factors be known beforehand, neither that they be equal since the methods discussed here make no use of Δ in their implementation.

Assumption (iii) prevents cancellation of effects within group-factors. Although interaction effects cannot be estimated by factor screening methods, their presence should not necessarily prevent the detection of important main effects, provided that the sum of the interaction effects within a group do not cancel out the sum of the main effects. For discussions and investigations of this see Kleijnen (1987) and references therein. Alternatively, interactions may be included in the model as if they were main effects (although this approach is not considered here).

The designs used when testing group-factors and individual factors in the 1st and 2nd stages respectively will be the multifactorial designs of Plackett and Burman (1946). These are orthogonal designs which consider all factors at two levels, and have the property that all main effects may be independently estimated with the smallest possible random error for a given number of trials. Such designs are given in [35], or may be derived from Hadamard matrices. Plackett and Burman designs exist only when the number of factors in the experiment plus one is divisible by 4. When this is not the case an extra 1, 2 or 3 runs will be required since the next largest design will be used. Assumption (iv) is included only to facilitate the ensuing analyses.

Assumption (v) enables the levels of factors to be chosen to ensure that no cancellations occur within a group. Such knowledge may be available from qualitative consideration of the system, but if this assumption doesn't hold then factor screening should still be feasible since the optimal group sizes are such that most groups should contain 0 or 1 effective factors, thus making cancellations impossible. Even when two or more effective factors do appear in the same group, cancellation will occur only in the pathological cases where the sum of the effects is approximately zero and thus masked by the experimental

error. Therefore, although necessary in theory, group screening should still be effective without this assumption. This is supported by Mauro and Smith (1982) who carried out an investigation into the performance of two-stage group testing when the directions of the effects are unknown, and concluded that even in the worst case (i.e. an equal number of positive and negative effects all with the same magnitude), group testing still performs well. An important difference in the implementation of factor screening is that two-sided significance tests would be used without Assumption (v), one-sided tests with Assumption (v).

Assumption (vii) is only made to simplify the following analysis. It is not necessary in practice that groups be of equal size, and in particular, if the optimal group size does not divide the population size exactly then groups may differ in size by one.

The work in this chapter is concerned with Assumption (vi) which assumes that the error variance is known. In practice one would generally not expect to know the variance and so, in the light of the above discussion, this would appear to be the main assumption preventing factor screening from being a fully practical method. In fact, the analysis in WAT uses t-tests when testing significance and so Assumption (vi) may be weakened to requiring an independent estimate of σ^2 for each stage, but since both first and second stage designs are nearly or completely saturated, the usual estimates are not available. If the number of factors plus one is not a multiple of 4 then some extra degrees of freedom are available, but this is clearly not a satisfactory solution, giving at best 3 and at worst 0 degrees of freedom to estimate σ^2. One possible approach would be to replicate each experiment as suggested by Kleijnen (1987), in which case the analysis would be as in WAT but with twice the number of runs. In the following sections two alternative methods are

presented which require fewer runs than fully replicated group testing and yet are effective at detecting the important factors. In Section 8.1 split-experiment factor screening is introduced which involves splitting the factors into two batches and performing separate experiments for each batch, replicating in the first stage only and "swapping" the estimates of σ^2 in the second stage. In Section 8.2 a simpler approach using theory from unreplicated factorial experimentation is given. Throughout these sections it will be assumed that model (8.1) and Assumptions (i)-(vii) hold, except that Assumption (vi) is replaced by the following.

(vi'.) The errors of observations are independent normal with *unknown* variance σ^2.

8.1 Split-Experiment Factor Screening

We seek a factor screening method which allows estimation of σ^2. One approach would be to carry on as usual but to replicate all observations in both stages. Savings in runs could be achieved however if the estimate of σ^2 used in the first stage could also be used in the second stage so that the latter would need no replication. However, if this were done then the usual t-statistic used in the second stage significance tests would no longer have a t-distribution since the factors tested in the second stage are from significant groups found in the first stage, and the first stage estimate of σ^2 would have been used to test this significance. *Split-experiment factor screening* overcomes this problem and will now be defined.

The N factors are divided into g groups of size k (optimum choice of k will be discussed later). The distinguishing feature of this method is that the groups themselves are split

into two equal-sized batches, and factors and group-factors are only ever tested with factors and group-factors from the same batch. In the first stage each batch of group-factors is tested in a Plackett and Burman design (so that two separate experiments are carried out in the first stage) with replication $r = 2$. This enables all group-factor main effects to be unbiasedly estimated and provides for both batches independent estimates of the error variance, s_1^2 and s_2^2 say. For each batch $g/2$ main effects plus a constant term are estimated using $g + 2$ runs and so s_1^2 and s_2^2 will both be based on $g/2 + 1$ degrees of freedom. In the second stage, for each batch, factors from significant groups are run in an experiment, again using Plackett and Burman designs (so that the second stage consists of two experiments, one for each batch), but this time without replication. These experiments allow the main effects of all factors in the second stage to be unbiasedly estimated. To test these factors for significance t-tests are again used, factors in the first batch using s_2^2 as the estimate of σ^2 in the denominator of the t-statistic, and factors in the second batch using s_1^2. The numerators are clearly independent of the denominators in this case and so under the null hypothesis the t-statistics will indeed have t-distributions, all with $g/2 + 1$ degrees of freedom. Those factors found to be significant are classified as effective.

The number of effective and ineffective factors declared significant and the number of runs required will now be studied for this method. Since split-experiment factor screening consists essentially of two separate factor screening experiments (one for each batch), the properties of this method will be similar to that in WAT. The distribution of the above statistics is more complex here however since the same estimates of σ^2 are used for both batches (although in reverse order) and so the results from the two batches are dependent. However, the average values can be found since the total number of runs is the sum of

the runs in the two batches and the number of effective and ineffective factors declared significant is the sum of those in the batches. The following work is similar to that in WAT, the main difference being in the power of the t-tests. Also, the mistakes in WAT pointed out by Gurnow (1965) have been avoided.

Consider the first stage. For each batch the group-factors are run in an experiment and tested for significance using t-tests. For a group-factor having mean effect μ, let $\pi_1(\mu)$ be the power of the t-test. Therefore, from Assumption (ii), $\pi_1(x\Delta)$ is the probability that a group containing x effective factors is declared significant in the first stage. Now both experiments in the first stage involve $g/2$ group-factors plus a constant term and use $2(g/2+1)$ runs. Therefore the variance of the estimate of the main effect of each group-factor is $\sigma^2/(g+2)$. The power of a t-test is given by a non-central t-distribution, and so if α_1 is the size of the t-tests in the first stage then the probability that a particular group containing x effective factors is declared significant is given by

$$\pi_1(x\Delta) = \text{prob}\left(t + \frac{x\Delta\sqrt{g+2}}{s} > t_{g/2+1;1-\alpha_1} \,\middle|\, t \sim t_{g/2+1} \;;\; s^2 \sim \frac{\sigma^2}{g/2+1}\chi^2_{g/2+1}\right). \tag{8.2}$$

Clearly the probability that an ineffective group-factor is declared significant is

$$\pi_1(0) = \alpha_1. \tag{8.3}$$

Tables for the non-central t-distribution include Pearson and Hartley (1954) and Lieberman and Resnikoff (1957). Alternatively, (8.2) can be computed using the algorithm of Cooper (1968).

Let π_1^* be the probability that a group-factor is declared significant. Then conditioning

on the number of effective factors in the group and using Assumption (i) gives

$$\pi_1^* = \sum_{x=0}^{k} \binom{k}{x} p^x q^{k-x} \pi_1(x\Delta) \tag{8.4}$$

Now consider the second stage. For each batch an experiment is run consisting of the factors from significant group-factors plus a constant term. Let n_1 and n_2 be the number of significant group-factors found in the first and second batch respectively. Then the number of runs made in the second stage for the jth batch ($j = 1, 2$) is $n_j k + 1$ (recall that there is no replication in the second stage experiments). Therefore the estimate of each factor main effect in the jth batch will have variance $\sigma^2/(n_j k + 1)$, and from Assumption (ii) will have mean Δ for effective factors and 0 for ineffective factors. The factors in each batch are tested for significance using t-tests, the estimate of σ^2 being s_2^2 for factors in the first batch and s_1^2 for the second (recall that s_1^2 and s_2^2 are the estimates of σ^2 from the first-stage experiments for the first and second batch respectively). For the jth batch ($j = 1, 2$), let $\pi_2(\mu, n_j)$ be the power of the t-test for a factor with mean effect μ (note that this power depends on n_j since this determines the number of runs made in the second stage). Then if α_2 is the size of the t-tests in the second stage, then the probability that an effective factor from a significant group in the jth batch is declared significant is

$$\pi_2(\Delta, n_j) = \text{prob}\left(t + \frac{\Delta\sqrt{n_j k + 1}}{s} > t_{g/2+1;1-\alpha_2} \,\middle|\, t \sim t_{g/2+1} \;;\; s^2 \sim \frac{\sigma^2}{g/2+1}\chi^2_{g/2+1} \right)$$

while the probability that an ineffective factor is declared significant is

$$\pi_2(0, n_j) = \alpha_2.$$

Having established the above properties, the performance of split-experiment factor screening may now be studied. Consider first the expected number of runs made, R_{se} say. In the first stage $2(g/2+1)$ runs are made for each batch, and in the second stage $n_j k+1$ runs are made for the jth batch ($j=1,2$). Therefore

$$R_{se} = 2g + n_1 k + n_2 k + 6.$$

Now the number of significant groups in the jth batch has the Binomial distribution

$$n_j \sim \text{Bin}(g/2, \pi_1^*) \qquad j = 1, 2, \tag{8.5}$$

and so

$$\begin{aligned} R_{se} &= 2g + gk\pi_1^* + 6 \\ &= 2g + N\pi_1^* + 6 \qquad \text{(using Assumption (vii)).} \end{aligned}$$

An upper bound and approximation may be obtained for R_{se} by noting that

$$\pi_1(x\Delta) \leq 1 \qquad \text{for } x = 1, 2, \ldots, k \tag{8.6}$$

and so from (8.4)

$$\begin{aligned} \pi_1^* &\leq q^k \pi_1(0) + \sum_{x=1}^{k} \binom{k}{x} p^x q^{k-x} \\ &= q^k \alpha_1 + (1 - q^k) \qquad \text{using (8.3)} \end{aligned}$$

Therefore,

$$\begin{aligned} R_{se} &\leq 2g + N\left(q^k\alpha_1 + (1-q^k)\right) + 6 \\ &= N\left(1 - (1-\alpha_1)q^k + \frac{2}{k}\right) + 6 \\ &= \hat{R}_{se} \qquad \text{say.} \end{aligned}$$

Now if Δ is large enough then the inequality in (8.6) will be almost an equality, and so $\hat{R}_{se}$ provides an approximation to the expected number of runs made. An approximation to the optimum group sizes for split-experiment group testing may therefore be obtained by minimizing $\hat{R}_{se}$. Allowing k to vary continuously, this is given by the solution of

$$-(1-\alpha_1)q^k \log q - \frac{2}{k^2} = 0. \tag{8.7}$$

Now when p is small,

$$q^k \approx 1 - kp$$

and

$$\log q \approx -p.$$

Substituting these into (8.7) gives

$$(1-\alpha_1)(1-kp)p - \frac{2}{k^2} \approx 0,$$

which, if p is small so that quadratic terms may be ignored, gives an approximation to the

optimal group size, $\hat{k}_{se}$ say, as

$$\hat{k}_{se} = \sqrt{\frac{2}{(1-\alpha_1)p}}.$$

This differs from the optimal group size in unreplicated factor screening only by the "2" in the numerator of the square root term. Table 8.1 shows $\hat{k}_{se}$ for various values of p with $\alpha_1 = 0.05$. These group sizes do not vary much over the likely range of choices of α_1.

p	0.2	0.15	0.1	0.05	0.04	0.03	0.02	0.01
k_{se}	3	4	5	6	7	8	10	15

Table 8.1: Optimum group sizes for split-experiment factor screening with $\alpha_1 = 0.05$

Consider now the number of effective factors which are correctly declared significant. Following the notation in WAT, let this be denoted by M_R. Then

$$M_R = M_R^{(1)} + M_R^{(2)} \tag{8.8}$$

where $M_R^{(j)}$ $(j = 1, 2)$ is the number of effective factors from the jth batch which are declared significant. Now n_j is the number of group-factors from the jth batch which are declared significant in the first stage, and for $i = 1, 2, \ldots, n_j$ let m_{ij} be the number of effective factors from the ith significant group in the jth batch which are declared significant. Then

$$M_R^{(j)} = m_{1j} + m_{2j} + \cdots + m_{n_j j} \qquad (j = 1, 2) \tag{8.9}$$

Now

$$\mathrm{E}(m_{ij}) = \sum_{x=0}^{k} \mathrm{E}(m_{ij} \mid i\text{th significant group contains } x \text{ effective factors})$$

$$\times \text{ prob}(i\text{th significant group contains } x \text{ effective factors}). \qquad (8.10)$$

Now effective factors in the second stage are declared significant with probability $\pi_2(\Delta, n_j)$, so that

$$\mathrm{E}(m_{ij} \mid x \text{ effective factors}) = x\pi_2(\Delta, n_j). \qquad (8.11)$$

The probability that a significant group contains x effective factors is given by

$$\begin{aligned} & \text{prob}(x \text{ effective factors} \mid \text{group significant}) \\ = \; & \frac{\text{prob}(\text{group significant} \mid x \text{ effective factors}) \times \text{prob}(x \text{ effective factors})}{\text{prob}(\text{group significant})} \\ = \; & \frac{\pi_1(x\Delta)\binom{k}{x} p^x q^{k-x}}{\pi_1^*}. \end{aligned} \qquad (8.12)$$

So substituting (8.11) and (8.12) into (8.10) gives

$$\mathrm{E}(m_{ij}) = \frac{\pi_2(\Delta, n_j)}{\pi_1^*} \sum_{x=0}^{k} x\pi_1(x\Delta) \binom{k}{x} p^x q^{k-x}.$$

Therefore, using (8.9) and conditioning on n_j gives

$$\mathrm{E}(M_R^{(j)} | n_j = n) = n\frac{\pi_2(\Delta, n)}{\pi_1^*} \sum_{x=0}^{k} x\pi_1(x\Delta) \binom{k}{x} p^x q^{k-x}$$

and so

$$\mathrm{E}(M_R^{(j)}) = \sum_{n=0}^{g/2} \mathrm{E}(M_R^{(j)} | n_j = n) \times \text{prob}(n_j = n)$$

$$= \frac{1}{\pi_1^*} \sum_{x=0}^{k} x\pi_1(x\Delta) \binom{k}{x} p^x q^{k-x} \sum_{n=0}^{g/2} n\pi_2(\Delta, n) \binom{g/2}{n} \pi_1^{*n}(1-\pi_1^*)^{g/2-n} \qquad \text{(using (8.5))}.$$

Therefore, from (8.8), the expected number of effective factors declared significant is

$$\mathrm{E}(M_R) = \frac{2}{\pi_1^*} \sum_{x=0}^{k} x\pi_1(x\Delta) \binom{k}{x} p^x q^{k-x} \sum_{n=0}^{g/2} n\pi_2(\Delta, n) \binom{g/2}{n} \pi_1^{*n}(1-\pi_1^*)^{g/2-n}. \qquad (8.13)$$

To obtain an approximation and lower bound to $\mathrm{E}(M_R)$, note first that the power $\pi_1(x\Delta)$ is the probability that a group containing x effective factors is declared significant, and so, since all effects are non-negative,

$$\pi_1(\Delta) \leq \pi_1(x\Delta) \qquad x = 1, 2, \ldots, k.$$

Substituting this into (8.13) gives

$$\mathrm{E}(M_R) \geq \frac{2}{\pi_1^*} \pi_1(\Delta) kp \sum_{n=0}^{g/2} n\pi_2(\Delta, n) \binom{g/2}{n} \pi_1^{*n}(1-\pi_1^*)^{g/2-n}. \qquad (8.14)$$

Note also that the power $\pi_2(\Delta, n)$ is the probability that an effective factor from a significant group is declared significant in the second stage where the number of runs in the experiment is $nk + 1$. Therefore, for $n = 1, 2, \ldots,$

$$\pi_2(\Delta, n) \geq \pi_2(\Delta, 1).$$

Substituting this into (8.14) gives

$$\begin{aligned} \mathrm{E}(M_R) &\geq \frac{2}{\pi_1^*}\pi_1(\Delta)kp\pi_2(\Delta,1)\frac{g}{2}\pi_1^* \\ &= Np\pi_1(\Delta)\pi_2(\Delta,1) \qquad \text{using Assumption (vii)} \end{aligned} \tag{8.15}$$

Now if both the powers in (8.15) are high and p is small, then this inequality should be almost an equality, and therefore (8.15) provides an approximation to the expected number of effective factors declared significant. Note that Np is the expected number of effective factors present, and therefore

$$\frac{\mathrm{E}(M_R)}{Np} \approx \pi_1(\Delta)\pi_2(\Delta,1)$$

is a measure of the efficiency of split-experiment factor screening for detecting effective factors.

This conclusion is similar to that in WAT for the case of known variance, except that the powers are different.

A similar argument to the above may be used for the number of ineffective factors which are incorrectly declared significant. Again following the notation in WAT, let this be denoted by M_U. Then

$$M_U = M_U^{(1)} + M_U^{(2)} \tag{8.16}$$

where $M_U^{(j)}$ is the number of ineffective factors declared significant in the jth batch. Let u_{ij} be the number of ineffective factors from the ith significant group in the jth batch

which are declared significant. Then

$$M_U^{(j)} = u_{1j} + u_{2j} + \cdots + u_{n_j j}. \tag{8.17}$$

Conditioning on the number of effective factors in the ith significant group,

$$\begin{aligned} \mathrm{E}(u_{ij}) = \sum_{x=0}^{k} \quad & \mathrm{E}(u_{ij} \mid i\text{th significant group contains } x \text{ effective factors}) \\ & \times \text{prob}(i\text{th significant group contains } x \text{ effective factors}). \end{aligned} \tag{8.18}$$

Now ineffective factors in the second stage are declared significant with probability α_2 so that

$$\mathrm{E}(u_{ij} \mid i\text{th significant group contains } x \text{ effective factors}) = (k-x)\alpha_2.$$

Substituting this and (8.12) into (8.18) we obtain

$$\mathrm{E}(u_{ij}) = \frac{\alpha_2}{\pi_1^*} \sum_{x=0}^{k} (k-x)\pi_1(x\Delta) \binom{k}{x} p^x q^{k-x}.$$

Then from (8.17),

$$\mathrm{E}(M_U^{(j)} | n_j = n) = n\frac{\alpha_2}{\pi_1^*} \sum_{x=0}^{k} (k-x)\pi_1(x\Delta) \binom{k}{x} p^x q^{k-x}$$

and so from (8.5),

$$\mathrm{E}(M_U^{(j)}) = \frac{\alpha_2}{\pi_1^*} \sum_{x=0}^{k} (k-x)\pi_1(x\Delta) \binom{k}{x} p^x q^{k-x} \sum_{n=0}^{g/2} n \binom{g/2}{n} \pi_1^{*n}(1-\pi_1^*)^{g/2-n}$$

$$= \frac{g\alpha_2}{2}\sum_{x=0}^{k}(k-x)\pi_1(x\Delta)\binom{k}{x}p^x q^{k-x}.$$

Therefore, from (8.16), the expected number of ineffective factors incorrectly declared significant is

$$\mathrm{E}(M_U) = g\alpha_2\sum_{x=0}^{k}(k-x)\pi_1(x\Delta)\binom{k}{x}p^x q^{k-x}.$$

8.2 Unreplicated Factor Screening

Consider the factor screening method described in WAT. The N factors are split into g groups of size k and the group-factors are tested in a Plackett and Burman design without replication. All group-factor main effects are therefore estimable, but since the design is saturated (i.e. the number of runs in the experiment is equal to the number of effects being estimated), no degrees of freedom are available to estimate σ^2. Similarly in the second stage, factors from significant groups are tested, also using an unreplicated Plackett and Burman design and so again leaving no degrees of freedom for error. However, methods for detecting significant factors in unreplicated factorials are available, see for example Daniel (1959), Zahn (1975) and Box and Meyer (1986). By applying such methods, factor screening without replication as in WAT could be used even when the error variance is unknown. The particular method to be considered here is due to Lenth (1989). It is suitable since it is simple and requires neither calculation of special confidence limits nor prior information or upperbound to the number of effective factors (although it is only powerful if the proportion of these is small). A summary of this method will now be given.

Consider an unreplicated factorial experiment involving m factors of which only a small number are expected to be effective. Denote the factor effects by $\kappa_1, \kappa_2, \ldots, \kappa_m$, and let

$c_1, c_2, \ldots, c_m$ be the corresponding estimates. Then the c_i will be independent normal variables with means κ_i and with equal variances, τ^2 say. Let

$$s_0 = 1.5 \times \text{median}|c_j|.$$

Then τ may be estimated by the *pseudo random error* (PSE):

$$PSE = 1.5 \times \text{median}(|c_j| \ : \ c_j < 2.5 s_0)$$

Lenth claims that, under the assumption of effect sparsity, the distribution of PSE is well approximated by a chi-square distribution with $m/3$ degrees of freedom, and so the significance of the effects may be tested using t-tests (the marginal dependence between PSE and c_i $(i = 1, 2, \ldots, m)$ should not be too great provided that m is not too small).

The justification of this method is that if all κ_i's were zero then the c_i's would be independent realizations of an $N(0, \tau^2)$ random variable C. Therefore s_0 would be a consistent estimate of $1.5 \times \text{med}|C| \approx 1.01\tau$. Further, since $\text{prob}(|C| > 2.5\tau) \approx .01$, PSE would be roughly consistent for 1.5 times the .495th quantile of $|C|$ which is approximately τ. Now if there is a small number of effective factors then PSE should be based on all of the ineffective factors and possibly a few of the smaller effective ones. Therefore PSE overestimates τ, but if the proportion of effective factors is small then the bias should not be great. In fact, in a monte-carlo study Lenth showed that PSE provides a reasonable estimate if the proportion of effective factors is less than 0.3 and m is greater than 7.

It is therefore proposed that the factor screening method in WAT as described at the beginning of this section be followed. At the end of each stage group-factors and factors

are tested for significance by t-tests using PSE as the estimator of σ in the denominator of the test statistic. In the first stage the t-tests will have $g/3$ degrees of freedom, while those in the second stage will have $nk/3$ where n is the number of groups declared significant in the first stage.

Since t-tests are used in this method, the analysis in WAT is appropriate (with degrees of freedom as above). All the results from WAT (noting the corrections of [12]) will hold approximately, the accuracy being determined by the closeness of PSE to a chi-square distribution. In particular, it is seen that the expected number of runs is approximately minimized by using groups of size $\hat{k}_{PSE}$ where

$$\hat{k}_{PSE} = \sqrt{\frac{1}{(1-\alpha_1)p}}$$

Table 8.2 shows $\hat{k}_{PSE}$ for various values of p with $\alpha_1 = 0.05$. These group sizes do not vary much over the likely range of choices of α_1.

p	0.2	0.15	0.1	0.05	0.04	0.03	0.02	0.01
$\hat{k}_{PSE}$	2	3	3	5	5	6	7	10

Table 8.2: Optimum group sizes for unreplicated factor screening using PSE with $\alpha_1 = 0.05$

If considering using this method it is however important to note that the requirement that the proportion of effective factors should be less than 0.3 (as discussed earlier) is in fact stricter than it appears. This is because in the first stage PSE is calculated for group-factors, which have a higher probability of being effective than the individual factors. Assuming that the above optimal group size $\hat{k}_{PSE}$ is used, the probability of a

group-factor being effective, p^* say, is

$$p^* = 1 - q^{((1-\alpha_1)p)^{-0.5}}.$$

Supposing that $\alpha_1 < 0.1$, then p^* will be less than 0.3 if $p < 0.1$. It is therefore suggested that this method will only be effective if $p < 0.1$. The simulation study in the following section confirms this.

8.3 Performance of Methods

This section discusses the likely properties of the two proposed methods under varying parameter values and presents the results of a simulation study which provides interesting insight and aids comparison of the methods.

First note that, whereas in group testing without errors there is a single main objective (to minimize the number of runs), in the presence of errors there are several objectives, mainly to minimize runs, to maximize the number of effective factors detected and to minimize the number of ineffective factors detected. These are of course conflicting requirements since an increase in the number of runs gives more accurate estimates and improves the power of the t-tests. Hence one method may be considered better than another only if it gives better results for all three objectives. More generally the practical implications of the experiment will determine which of the three objectives are the most important.

Let us now intuitively examine the likely effects of varying parameter values for split-experiment factor screening. Consider first changes in the size of effective factors, Δ.

Clearly an increase in Δ will lead to better detection of effective factors. A small increase in the expected number of runs would also be expected since effective groups are less likely to be missed in the first stage and hence more likely to be tested in the second. This increase in runs is not however to be considered as undesirable since any savings in this sense are due only to mistakes in the first stage. The number of ineffective factors declared significant should not be directly affected by an increase in Δ. Now consider the effects of changes in p, the probability of a factor being effective. Clearly as p gets smaller there will be fewer effective factors, thus leading to a reduction in the number of runs. However, this decrease in runs will reduce the power of t-tests and thus adversely affect the detection of effective factors. So a reduction in p leads to savings in runs but at the expense of power.

For unreplicated factor screening using PSE to estimate σ, the effects of varying Δ are not so clear. This is because changes in Δ are likely to affect the performance of PSE as an estimator of σ. This conflicts with the fact that larger effects are better detected for any fixed estimate of σ. The effects of varying p discussed above for split-experiment factor screening are also relevant here, but made more complex by the fact that PSE becomes a more accurate estimator of σ as p decreases, and as mentioned earlier, is likely to become poor for p greater than 0.1.

Tables 8.3 and 8.4 present the results of a simulation study of factor screening using the proposed methods. The simulated experiments involved approximately 100 factors (the exact number of factors was determined by the grouping scheme and was chosen to allow equal-sized groups and batches), with test sizes $\alpha_1 = \alpha_2 = 0.05$ and use the optimum group sizes given in Tables 8.1 and 8.2. Each simulation involved 10,000 runs. This gave

estimates for the expected number of runs with a standard error of about 1% for split-experiment factor screening and slightly less for the unreplicated case. The tables give a summary of the results for various values of Δ and p. All of the intuitive comments made above are reflected in these results.

		$\Delta = 0.75\sigma$	$\Delta = \sigma$	$\Delta = 2\sigma$	$\Delta = 3\sigma$
$p = 0.2$	$\overline{R}$	1.237	1.237	1.238	1.240
	$\overline{M}_R$	0.975	0.998	1.000	1.000
	$\overline{M}_U$	0.019	0.020	0.019	0.019
$p = 0.1$	$\overline{R}$	0.884	0.897	0.899	0.899
	$\overline{M}_R$	0.914	0.987	0.998	1.000
	$\overline{M}_U$	0.018	0.018	0.018	0.019
$p = 0.05$	$\overline{R}$	0.676	0.696	0.699	0.696
	$\overline{M}_R$	0.792	0.938	0.973	1.000
	$\overline{M}_U$	0.012	0.013	0.013	0.013
$p = 0.01$	$\overline{R}$	0.323	0.350	0.382	0.382
	$\overline{M}_R$	0.506	0.714	0.919	1.000
	$\overline{M}_U$	0.006	0.007	0.008	0.009
$\overline{R}$ = Average number of runs per factor $\overline{M}_R$ = Average proportion of effective factors detected $\overline{M}_U$ = Average proportion of ineffective factors detected					

Table 8.3: Results of simulation study of split-experiment factor screening. Number of factors ≈ 100, $\alpha_1 = \alpha_2 = 0.05$.

The tables show that both methods can lead to detection of effective factors using a relatively small number of runs. As was expected, unreplicated factor screening using PSE performs poorly when p is greater than about 0.1. For smaller values of p however it performs well, detecting factors with effects larger than two standard deviations very efficiently and using very few runs. Split-experiment factor screening, while using more runs than the unreplicated case, is generally better at detecting effective factors, particularly for larger p and smaller Δ.

We would perhaps expect split-experiment factor screening to be uniformly better at detecting effective factors than the unreplicated method since it uses exact t-tests and more

		$\Delta = 0.75\sigma$	$\Delta = \sigma$	$\Delta = 2\sigma$	$\Delta = 3\sigma$
$p = 0.2$	$\overline{R}$	0.875	0.892	0.897	0.899
	$\overline{M}_R$	0.074	0.179	0.229	0.223
	$\overline{M}_U$	5e-05	4e-05	0.003	0.003
$p = 0.1$	$\overline{R}$	0.624	0.647	0.652	0.652
	$\overline{M}_R$	0.713	0.945	0.993	0.991
	$\overline{M}_U$	0.002	0.005	0.008	0.008
$p = 0.05$	$\overline{R}$	0.403	0.450	0.467	0.469
	$\overline{M}_R$	0.586	0.894	0.989	0.990
	$\overline{M}_U$	0.004	0.006	0.007	0.007
$p = 0.01$	$\overline{R}$	0.171	0.195	0.231	0.232
	$\overline{M}_R$	0.289	0.553	0.988	0.993
	$\overline{M}_U$	0.002	0.002	0.003	0.003
$\overline{R}$ = Average number of runs per factor $\overline{M}_R$ = Average proportion of effective factors detected $\overline{M}_U$ = Average proportion of ineffective factors detected					

Table 8.4: Results of simulation study of factor screening using PSE to estimate σ. Number of factors ≈ 100, $\alpha_1 = \alpha_2 = 0.05$.

runs. This is not the case however, when p is small the unreplicated method is better at detecting effects of 2σ. This is explained by the grouping procedure. For example, when p is 0.01, split-experiment factor screening partitions the population into two batches of three groups. Since first stage tests are replicated twice for this method, each group-factor is therefore involved in six runs. In the unreplicated case however, the population is partitioned into ten groups and so each group is involved in ten runs. These extra runs lead to better detection of effective groups, despite using only an approximation to the t-test. This advantage disappears however as Δ increases and the power of the true t-test improves.

For both methods the proportion of ineffective factors incorrectly detected was very small (2% in the worst case). The unreplicated method in particular made very few errors of this type.

Part V

ROBUST CIRCUIT DESIGN — AN EXAMPLE

Chapter 9

An Application of Group Testing to Robust Circuit Design

Within the Engineering Design Centre at City University a research project is underway concerned with the design of electronic circuits. This is a collaborative project with Mentor Graphics UK Ltd, the aim of which is to apply robust engineering design (RED) principles to circuit design by developing statistical modules which can be directly bolted on to the design software, thus allowing the designer to make use of the innovations arising in RED within the same menu driven environment. The software system used is the Mentor Graphics simulator Accusim. The statistical modules implemented include experimental design, a regression package and factor plots. These modules allow the designer to model the relationship between the inputs and outputs of a circuit, and thus perform optimization and sensitivity analysis more efficiently than by using traditional Monte-Carlo analysis, which can be prohibitively expensive in computer time for large problems. For further reading on this project see [1], [2], [3] and [46].

This chapter describes an application of group testing to circuit tolerance design. In this problem it is assumed that the topology of the circuit and nominal values for each component in the circuit have already been chosen by the designer. The aim is to reduce the variation in the outputs of the circuit by reducing the variation in the inputs. This can be done by using more expensive components which have tighter tolerances. By performing statistical experiments, the circuit may be modeled and the model then used to choose which components to upgrade. Quadratic effects and two-way interactions are common and so the regression package uses 3^k designs, which allow up to 40 variables, and Latin Hypercube Sampling (L.H.S.) which allows any number of variables and uses $2N+10$ tests where N is the number of variables. However, the simulation time is a rapidly increasing function of circuit size and so experiments on large circuits are computationally expensive. Moreover, it often turns out that the outputs of a circuit are reasonably robust to variation in most of the input parameters. In such cases group testing may be useful to screen out the components to which the outputs are most sensitive. Further experimentation may then be concentrated on these important inputs.

From a group testing point of view there are two main advantages of using electronic circuits as a test bed for search algorithms. The first is that the relationship between the inputs and outputs of a circuit is generally very complex and many of the assumptions required for group testing do not hold, e.g. interactions are present, effects are non-linear, and the likely directions of the effects of components are unknown. Electronic circuits therefore allow an investigation of the performance of group testing algorithms with non-standard problems. The second advantage is that the circuit to be considered has been previously investigated in statistically designed experiments and the relationship between

the inputs and outputs previously modeled. Although no information from these analyses was used in the group testing experiments, the accuracy of the results from group testing may be compared to those from larger experiments.

9.1 The Circuit

The circuit used in this example, shown in Figure 9.1, controls the joystick on an electric wheelchair. The circuit consists of 44 components and 7 outputs of which 3 were of interest to the designer, namely X2-IC3, X1-IC4 and X2-IC4 which correspond to the left and right coordinates of the position of the joystick, and a test signal. Let the components be labeled $C_1, C_2, \ldots, C_{44}$. It is known from previous analysis of the circuit that two components, C_{43} and C_{44}, are of particular importance. To illustrate this a main effects model was fitted for each output of interest with each factor (component) taking two levels. These levels were taken to be the nominal value of the component (as specified by the designer), and the nominal value plus 20%. Figure 9.2 shows the estimates of the main effects for each component from the fitted models. The plots show that only five components have a non-zero effect on any of the outputs. As already mentioned, the components which have the largest effects are C_{43} and C_{44} which influence outputs X2-IC4 and X2-IC3 respectively. Components C_1 and C_3 have a non-zero, though smaller influence on outputs X2-IC3 and X2-IC4, and components C_2 and C_3 influence output X1-IC4. Two other important features of the circuit are that the effects of important factors do not all have the same sign (as can be seen in Figure 9.2), and interactions are present.

Figure 9.1: Electronic circuit for controlling the joystick on an electric wheelchair

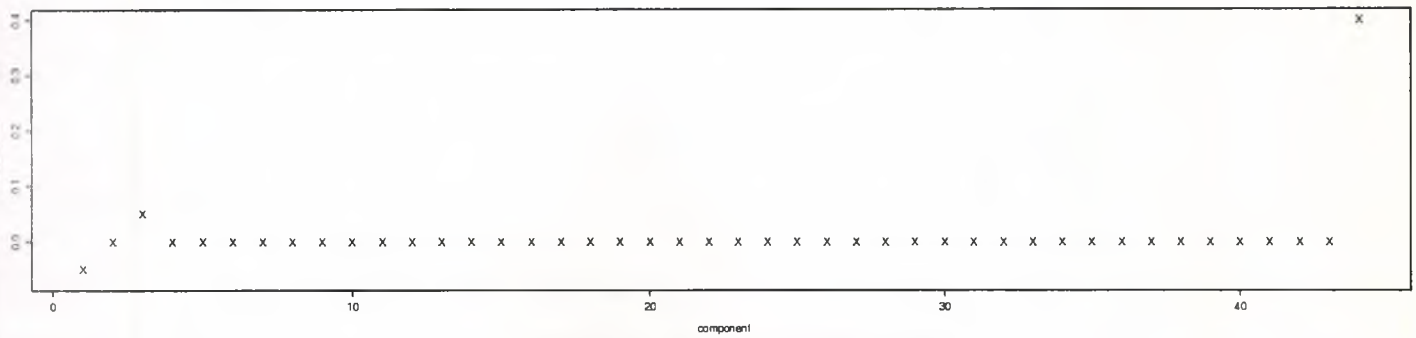

Figure 9.2 (a). Main effect plots of components for output X2-IC3

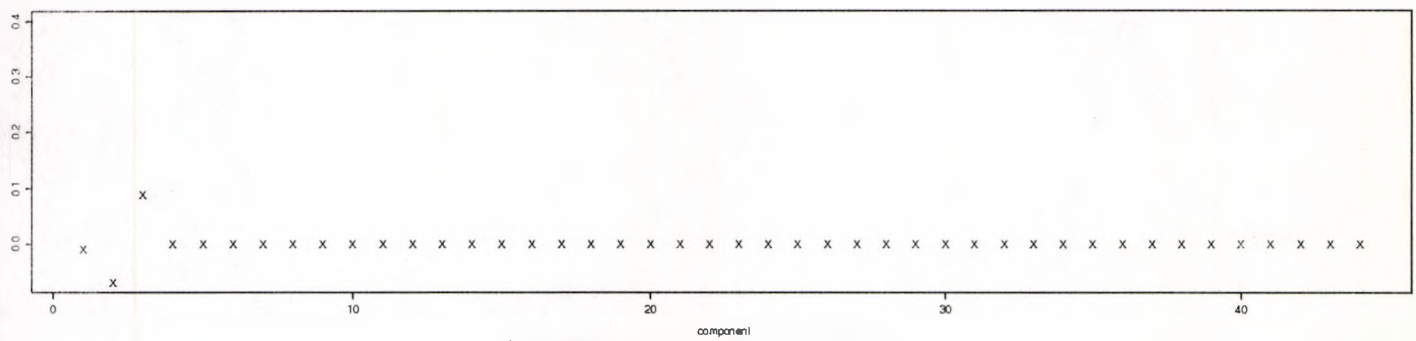

Figure 9.2 (b). Main effect plots of components for output X1-IC4

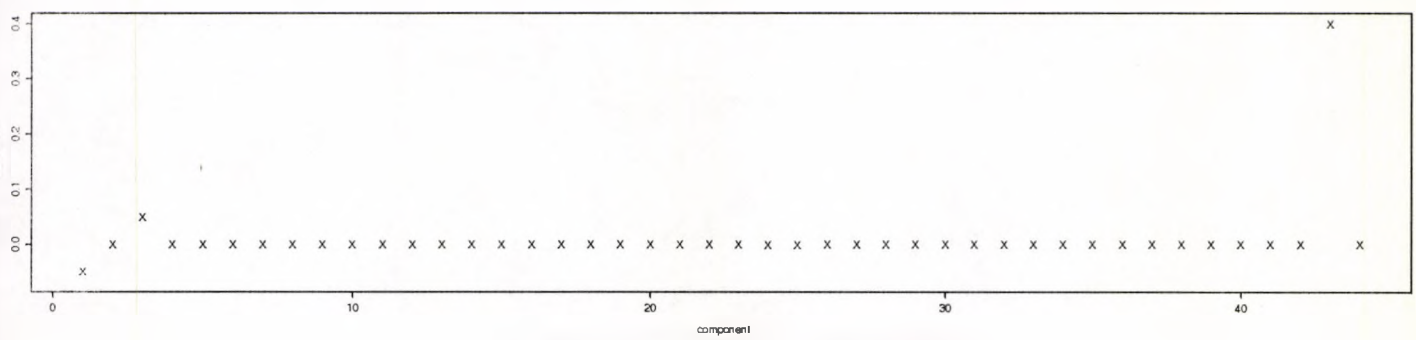

Figure 9.2 (c). Main effect plots of components for output X2-IC4

9.2 Group Testing Experiment

The circuit shown in Figure 9.1 was used as a test-bed for group testing. Two-stage pooling algorithms were used which involved partitioning the components into group-factors taking two levels, a lower level when all components in the group were assigned their nominal values, and an upper level when all components in the group were assigned their nominal level plus 20%. The main effects of the group-factors were estimated for each output, and the individual components within groups having a non-zero effect for at least one output were tested in the second stage.

Since no prior information was assumed concerning the proportion of important components, group testing experiments were performed using various values for the group size k. The groups were formed consecutively in such a way that k varied from group to group by at most one. So for example, with $k = 3$ the circuit was partioned into 15 groups, $G_1, G_2, \ldots, G_{15}$ say, where

$$\begin{aligned} G_1 &= (C_1, C_2, C_3) \\ G_2 &= (C_4, C_5, C_6) \\ &\vdots \\ G_{14} &= (C_{40}, C_{41}, C_{42}) \\ G_{15} &= (C_{43}, C_{44}). \end{aligned}$$

The plots in Figures 9.3–9.6 show the estimates of the main effects of the group-factors for each output and for various values of k. It is seen that, in each case, one group-factor was found to be important for outputs X2-IC3 and X2-IC4, but none for output X1-IC4.

The components within the effective group were therefore tested in a second stage and Figure 9.2 shows that these tests revealed components C_{43} and C_{44} to be important for outputs X2-IC4 and X2-IC3 respectively.

Note however that none of the group-factors containing the components C_1, C_2 and C_3 were found to be important in the first stage and so these components were not detected by group testing. It would seem, from Figure 9.2, that for the outputs X2-IC3 and X2-IC4 this was due to cancellation of equal and opposite effects. For output X1-IC4 however, the effects of components C_2 and C_3 are not of the same magnitude and so it would seem that an interaction was present which cancelled the effects of C_2 and C_3 when they were varied together to their upper level.

Table 9.1 shows the number of tests used by the two-stage pooling algorithms with groups of size k.

k	3	5	7	9
Number of tests	18	14	13	14

Table 9.1: Number of tests using two-stage pooling algorithm

9.3 Conclusion

The example given in this chapter shows the potential usefulness of group testing while also illustrating the type of errors which may arise. The two most important components were detected using relatively few tests. In practice, further experimentation might be considered for these components in order to model them in greater detail. Table 9.1 shows that even with this further experimentation, group testing methods can lead to experimentation using much fewer runs than the methods currently being used. It should be

noted however that these savings in runs are a result of the small proportion of important components in the circuit. Group testing does not therefore offer an alternative to the 3^k or L.H.S. designs, but rather a preliminary method which in certain circumstances may be used to screen out the important components and therefore make further experimentation more efficient. It was also seen however that the presence of interactions and components having equal and opposite effects can lead to failure to detect non-zero effects, and care must therefore be taken when considering using these methods.

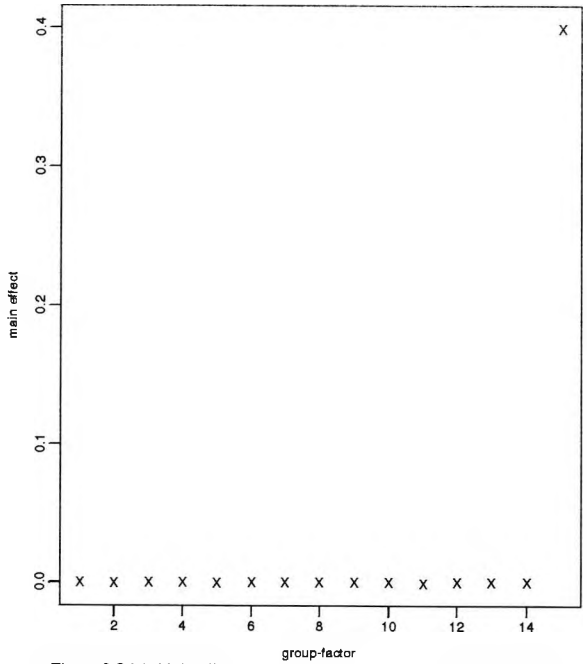

Figure 9.3 (a). Main effect plots of group-factors for output X2-IC3 with k=3

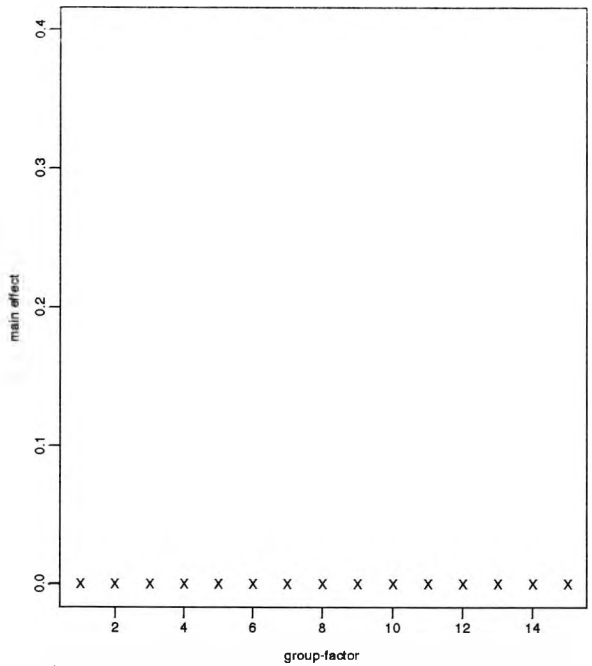

Figure 9.3 (b). Main effect plots of group-factors for output X1-IC4 with k=3

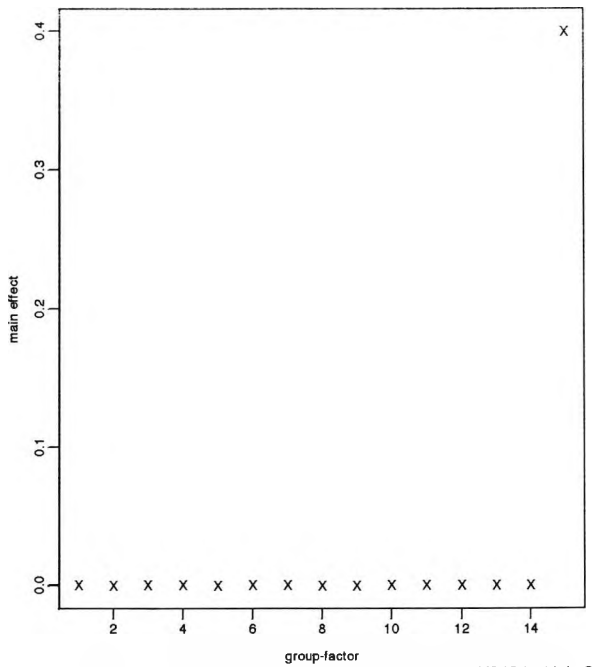

Figure 9.3 (c). Main effect plots of group-factors for output X2-IC4 with k=3

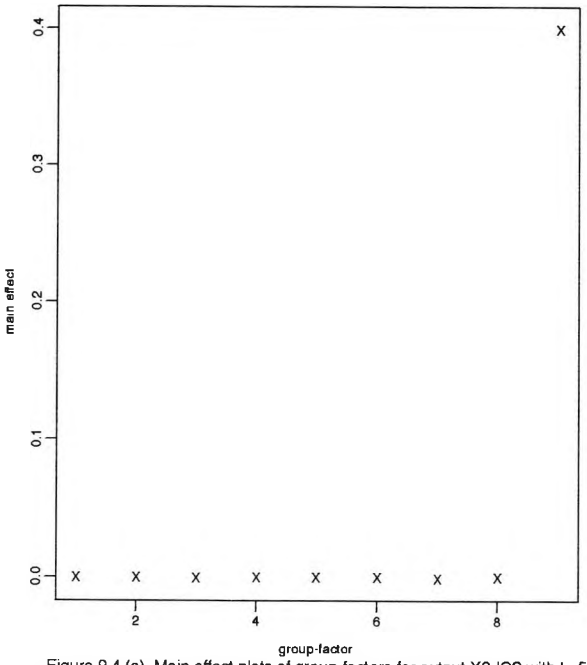

Figure 9.4 (a). Main effect plots of group-factors for output X2-IC3 with k=5

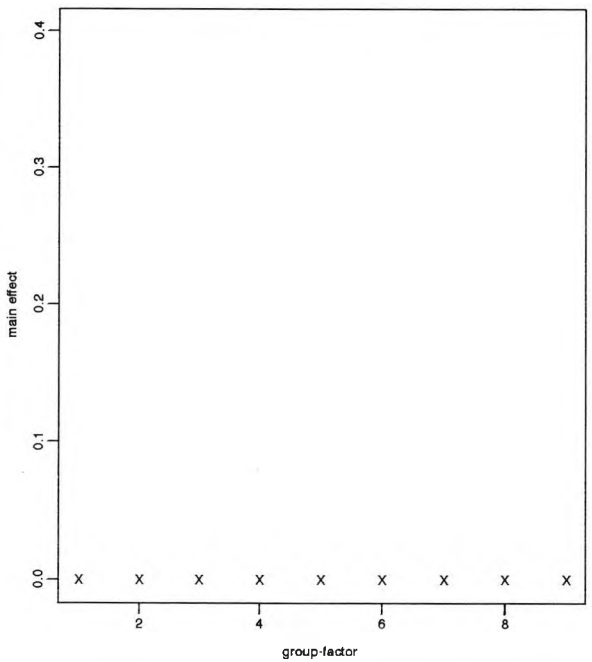

Figure 9.4 (b). Main effect plots of group-factors for output X1-IC4 with k=5

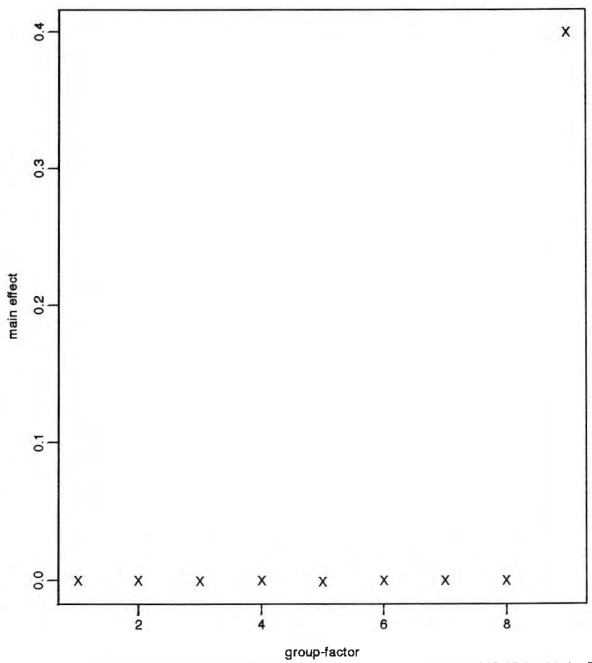

Figure 9.4 (c). Main effect plots of group-factors for output X2-IC4 with k=5

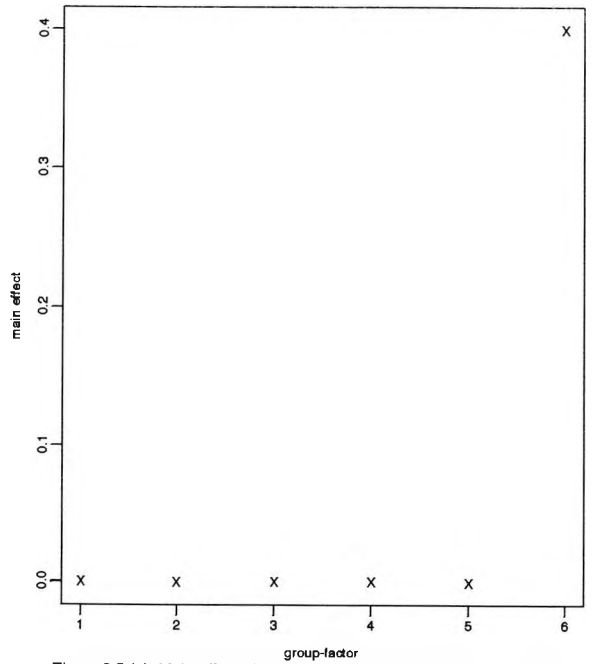

Figure 9.5 (a). Main effect plots of group-factors for output X2-IC3 with k=7

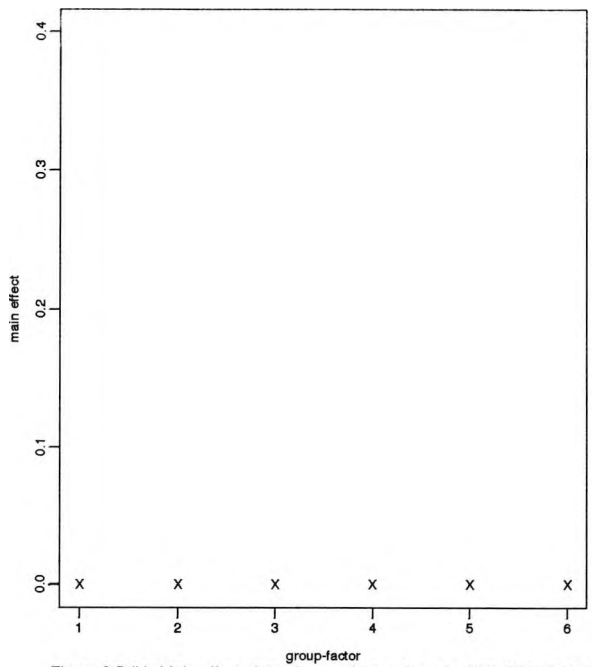

Figure 9.5 (b). Main effect plots of group-factors for output X1-IC4 with k=7

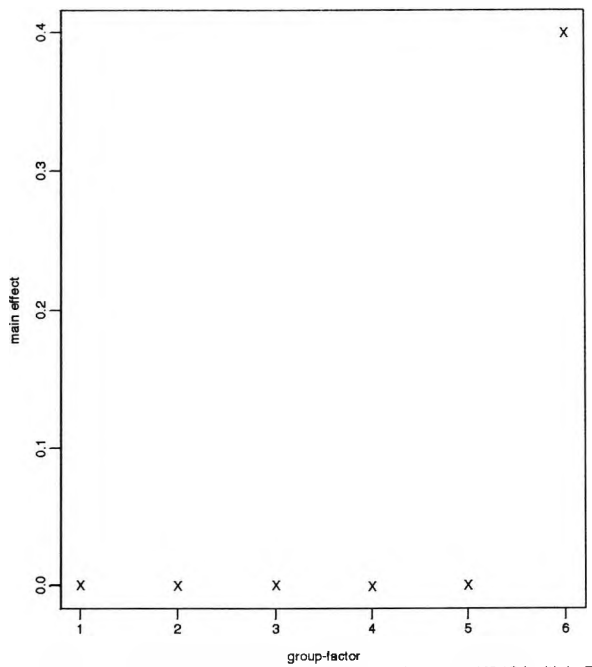

Figure 9.5 (c). Main effect plots of group-factors for output X2-IC4 with k=7

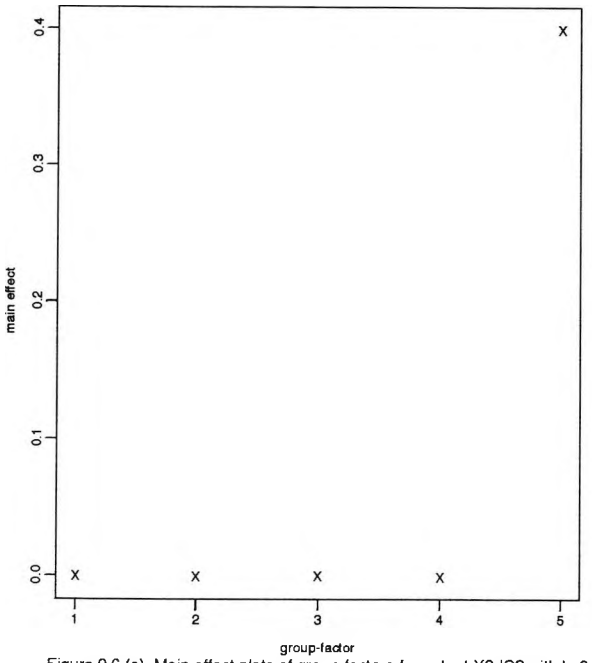

Figure 9.6 (a). Main effect plots of group-factors for output X2-IC3 with k=9

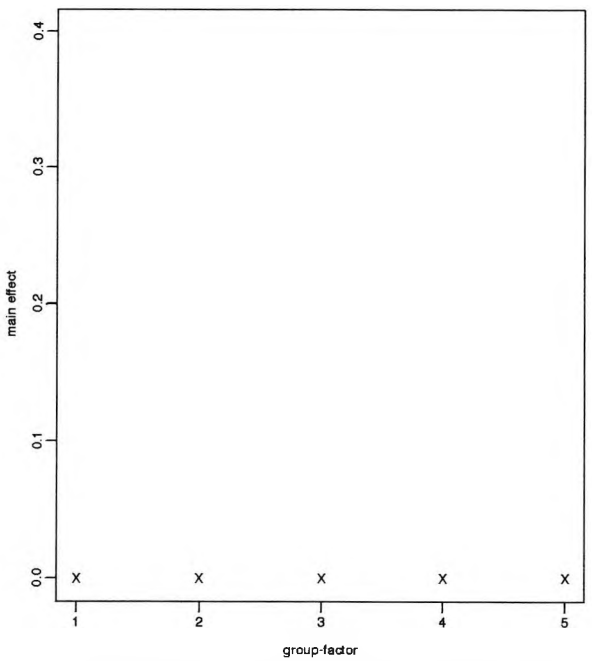

Figure 9.6 (b). Main effect plots of group-factors for output X1-IC4 with k=9

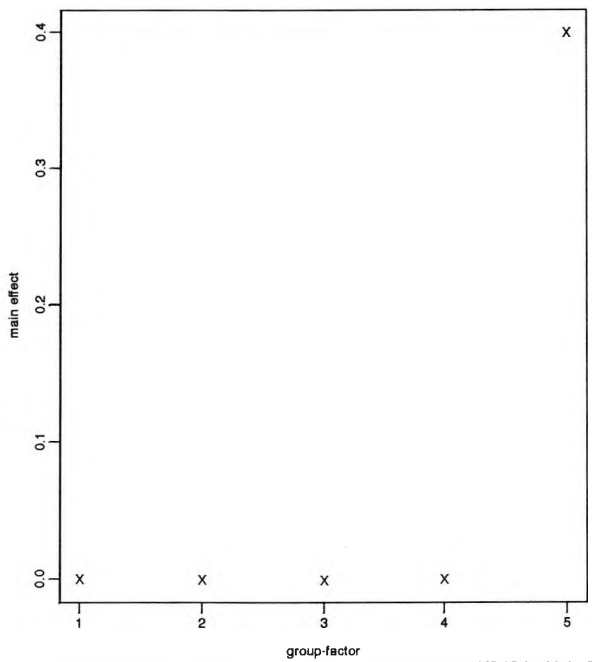

Figure 9.6 (c). Main effect plots of group-factors for output X2-IC4 with k=9

Bibliography

[1] Bates, R.A. (1991). Strategies for circuit optimization: a review. *Technical Report 32*, Engineering Design Centre, City University, London.

[2] Bates, R.A., Buck, R.J. and Wynn, H.P. (1991). Robust circuit design: an example. *Technical Report 29*, Engineering Design Centre, City University, London.

[3] Bates, R.A., Buck, R.J. and Wynn, H.P. (1992). Network partitioning and response modelling for robust engineering design. *Technical Report 42*, Engineering Design Centre, City University, London.

[4] Berger, T., Mehravari, N., Towsley, D. and Wolf, J.H. (1984). Random multiple-access communication and group testing. *I.E.E.E. Transactions on Communications*, **COM-32**, 769–779.

[5] Box, G.E.P. and Meyer, R.D. (1986). An analysis for unreplicated fractional factorials. *Technometrics*, **28**, 11–18.

[6] Cooper, B.E. (1968). Algorithm AS5. The integral of the non-central t-distribution. *Applied Statistics*, **17**, 193–194.

[7] Czyzowicz, KJ., Lakshmanan, K.B. and Pelc, A. (1991). Searching with a forbidden lie pattern in Responses. *Information Processors Letters*, **37**, 127–132.

[8] Daniel, C. (1959). Use of half-normal plots in interpretating factorial two level experiments. *Technometrics*, **1**, 311–341.

[9] Dorfman, R. (1943). The detection of defective members of large populations. *Annals of Mathematical Statistics*, **14**, 436–440.

[10] Glassey, C.R. and Karp, R.M. (1976). On the optimality of Huffman trees. *SIAM Journal of Applied Mathematics*, **31**, 368–378.

[11] Graff, L.E. and Roeloffs, R. (1974). A group testing procedure in the presence of test error. *Journal of the American Statistical Association*, **69**, 159–163.

[12] Gurnow, R.N. (1965). A note on G.S. Watson's paper "A study of the group screening method". *Technometics*, **7**, 444-446.

[13] Harrison, M.C. (1971). Implementation of the substring test by hashing. *Communications of the Association for Computing Machinery*, **14**, 777-779.

[14] Hayes, J.H. (1978). An adaptive technique for local distribution. *I.E.E.E. Transactions on Communications*, **COM-26**, 1178–1186.

[15] Hill, R. and Karim, J.P. (1992) Searching with lies: the Ulam problem. *Discrete Mathematics*, **106/107**, 273–283.

[16] Hu, T.C. and Tucker, A.C. (1971). Optimum computer search trees and variable-length alphabetic codes. *SIAM Journal of Applied Mathematics*, **21**, 514–532.

[17] Hunter, W.G. and Mezaki, R. (1964). Catalyst selection by group-testing. *Industrial and Engineering Chemistry*, **56**, 38–41.

[18] Hwang, F.K. (1976). An optimum nested procedure in binomial group testing. *Biometrics*, **32**, 939–943.

[19] Hwang, F.K. and Mallows, C.L. (1979). Some realizability theorems in group testing. *SIAM Journal of Applied Mathematics*, **37**, 396–400.

[20] Hwang, F.K., Pfeifer, C.G. and Enis, P. (1981). An optimal hierarchical procedure for a modified binomial group-testing problem. *Journal of the American Statistical Association*, **76**, 947–949.

[21] Kleijnen, J.P.C. (1987). Review of random and group-screening designs. *Communications in Statistics — Theory and Methods*, **16**, 2885–2900.

[22] Knuth, D. *The Art of Computer Programming*, vol. 3. Addison-Wesley, London, 1973.

[23] Lenth, R.V. (1989). Quick and easy analysis of unreplicated factorials. *Technometrics*, **31**, 469–473.

[24] Lieberman, G. and Resnikoff, G. *Tables of the Non-Central t-Distribution*. Stanford University Press, 1957.

[25] Mauro, C.A. and Smith, D.E. (1982). The performance of two-stage group screening in factor screening experiments. *Technometrics*, **24**, 325–330.

[26] Monzon, O.T., Paladin, F.J.E., Dimaandal, E., Balis, A.M., Samson, C. and Mitchell, S. (1992). Relevance of antibody content and test format in HIV testing of pooled sera. *AIDS*, **6**, 43–48.

[27] O'Geran, J.H., Wynn, H.P. and Zhiglyavsky, A.A. (1991). Search. *Acta Applicandae Mathematicae*, **25**, 241–276.

[28] O'Geran, J.H., Wynn, H.P. and Zhiglyavsky, A.A. (1992). Search and observer logics. *Proceedings of the Fifth Purdue Symposium on Statistical Decision Theory and Related Topics.*

[29] O'Geran, J.H., Wynn, H.P. and Zhiglyavsky, A.A. (1993a). Mastermind as a test-bed for search algorithms. *Chance*, **6**, 31–37.

[30] O'Geran, J.H., Wynn, H.P. and Zhiglyavsky, A.A. (1993b). Renyi type randomization theory for search length upperbounds. *Acta Applicandae Mathematicae* (to appear).

[31] Ottieno, J.A.M. and Patel, M.S. (1984). Two-stage group-screening designs with unequal a-prior probabilities. *Communications in Statistics — Theory and Methods*, **13**, 761–779.

[32] Patel, M.S. (1962). Group-screening with more than two stages. *Technometrics*, **4**, 109–217.

[33] Pearson, E.S. and Hartley, H.O. *Biometrika Tables for Statisticians*, vol. 1. Cambridge University Press, 1954.

[34] Pfeifer, C.G. and Enis, P. (1978). Dorfman-type group testing for a modified binomial model. *Journal of the American Statistical Association*, **73**, 588–592.

[35] Plackett, R.L. and Burman, J.P. (1946). The design of optimum multifactorial experiments. *Biometrika*, **33**, 305–325.

[36] Smith, D.E. (1973). Requirements of an "optimizer" for computer simulations. *Naval Research Logistics Quarterly*, **20**, 161–179.

[37] Sobel, M. (1968). Binomial and hypergeometric group testing. *Studia Scientiarum Mathematicarum Hungarica*, **3**, 19–42.

[38] Sobel, M. and Groll, P.A. (1959). Group testing to eliminate efficiently all defectives in a binomial sample. *Bell System Technical Journal*, **38**, 1179–1252.

[39] Sobel, M. and Groll, P.A. (1966). Binomial group testing with an unknown proportion of defectives. *Technometrics*, **8**, 631–656.

[40] Sterret, A. (1957). On the detection of defective members of large populations. *Annals of Mathematical Statistics*, **28**, 1033–1036.

[41] Thomas, J., Pasternack, B.S., Vacirca, S.J. and Thompson, D.L. (1973). Application of group-testing procedures in radiological health. *Health Physics*, **25**, 259–266.

[42] Thompson, K.H. (1962). Estimation of the proportion of vectors in a natural population of insects. *Biometrics*, **18**, 568–578.

[43] Ulam, S.M. (1976). *Adventures of a Mathematician*, p281. Scribner, New York.

[44] Watson, G.S. (1961). A study of the group screening method. *Technometrics*, **3**, 371-388.

[45] Wolf, J.K. (1985). Born again group testing: multiaccess communications. *I.E.E.E. Transactions on Information Theory*, **IT-31**, 185–191.

[46] Wynn, H.P., Buck, R.J. and Bates, R.A. (1992). A statistical bolt-on for robust circuit design. *Technical Report 61*, Engineering Design Centre, City University, London.

[47] Zahn, D.A. (1975). Modifications of and revised critical values for the half-normal plot. *Technometics*, **17**, 189–200.