*IIE Transactions* (2004) **36**, 481–493

ISSN: 0740-817X print / 1545-8830 online
DOI: 10.1080/07408170490426206

# An upper bound on the weight-balanced testing procedure with multiple testers

IRAD BEN-GAL

*Department of Industrial Engineering, Tel Aviv University, Ramat Aviv, Tel Aviv, 69978, Israel*
*E-mail: bengal@eng.tau.ac.il*

Received July 2002 and accepted November 2003

This paper presents the performance of the Weight-Balanced Testing (WBT) algorithm with multiple testers. The WBT algorithm aims to minimize the expected number of (round of) tests and has been proposed for coding, memory storage, search and testing applications. It often provides reasonable results if used with a single tester. Yet, the performance of the WBT algorithm with multiple testers and particularly its upper bound have not been previously analyzed, despite the large body of literature that exists on the WBT algorithm, and the recent papers that suggest its use in various testing applications. Here we demonstrate that WBT algorithm with multiple testers is far from being the optimal search procedure. The main result of this paper is the generalization of the upper bound on the expected number of tests previously obtained for a single-tester WBT algorithm. For this purpose, we first draw an analogy between the WBT algorithm and alphabetic codes; both being represented by the same $Q$-ary search tree. The upper bound is then obtained on the expected path length of a $Q$-ary tree, which is constructed by the WBT algorithm. Applications to the field of testing and some numerical examples are presented for illustrative purposes.

## 1. Introduction

In this paper we analyze the Weight-Balanced Testing (WBT) algorithm with multiple testers. We start by exemplifying the sequential search problem as presented in Ahlswede and Wegner (1987) and Yeung (1991). Consider a sequential input-output system (e.g., an electronic device) where the components are connected serially, such that the output of the $i$th component is the input for the $(i + 1)$th component. Figure 1 presents such a system with $n$ components linked serially from left to right. The premise is that the overall system output is being continuously monitored. Thus, upon first detecting an erroneous system output, it is reasonable to assume that there is exactly one component which is malfunctioning (we then term this component as MF). Each component has a certain (relative) reliability which is expressed in terms of its probability to be the MF. We denote this probability by $P_i, i = 1, \ldots, n$. Since the components are linked serially, a correct signal from the output of a component indicates that the MF is to its right, while an erroneous output indicates that the MF is to its left (including the inspected component itself). The problem is to find the MF as quickly as possible in order to repair or replace it, or, equivalently, the problem is to design a testing procedure that yields the lowest expected number of tests to indicate the MF. The expected number of tests is defined over the probabilities of the components to be the MF. When considering multiple testers, one aims to minimize the expected number of rounds of tests: a number which is proportional to the expected search time. In this paper, we use the term "number of tests" also for a multiple-testers search, although we actually refer to the number of rounds of tests. Note that such a search problem is applicable to different areas. Lipman and Abrahams (1995) considered a search for a defective segment in a pipeline with a finite number of segments linked together. The pipeline can be tested for defects only at a link. A test of a link establishes whether the defect is on the left or on the right of that link. He *et al*. (1996) and Herer and Raz (2000) considered a similar search problem of a defective product in a lot produced by a process with a constant failure rate. We consider the same problem in the numerical example in Section 5.

Considering the above search problem, we investigate the performance of the WBT algorithm with multiple testers. The WBT algorithm has been extensively proposed in the past for coding, memory storage, search and testing applications. The WBT algorithm is based on the simple principle of successive partitions of the search set into equiprobable subsets, i.e., subsets having an equal probability to contain the searched item. At each step of the algorithm, the subset which contains the searched item is partitioned in to equiprobable sub-subsets and the process repeats itself until the searched item is found. Intuitively, such a greedy principle seems appealing since it is simple to apply, provides an efficient search at each step individually and under certain conditions, attains the entropy lower bound on the

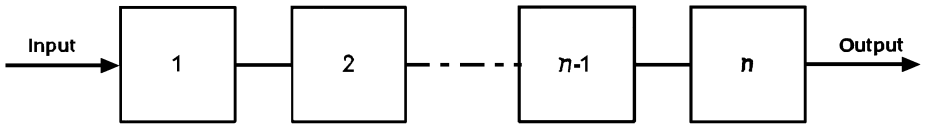


**Fig. 1.** A serial input-output system with $n$ components.

expected number of tests. However, as shown here, the WBT algorithm with multiple testers is far from being optimal. This phenomenon is sometimes overlooked, mainly when the WBT algorithm is applied to areas which are not related to coding. In fact, the performance of the WBT algorithm with multiple testers and particularly the upper bound on the expected number of tests have not been analyzed, despite the extensive literature that exists on the single-tester WBT algorithm. In this paper, we use the equivalence between testing procedure and prefix-free codes to derive an upper bound on the expected search length of the WBT algorithm with multiple testers. The search length of the WBT algorithm is equivalent to the expected code length represented by the different paths of a WBT search tree. We then compare our results to optimal testing procedures that are known from coding theory.

The rest of the paper is organized as follows. Section 2 provides a literature review. As seen, most of the mentioned papers are related to a single-tester WBT algorithm. Section 3 describes the WBT algorithm testing procedure. It addresses the analogy between the WBT algorithm and prefix-free codes, both represented by a WBT tree, and provides some known coding results. Section 4 obtains the upper-bound on the expected number of tests of the WBT algorithm with multiple testers. Section 5 presents analytic and numeric examples related to an industrial testing application. Section 6 concludes the paper.

## 2. Literature review

The considered testing problem has various forms depending on its area of application. The use of weight-balanced trees for searching applications related to computer and communication networks is very popular (Blum and Mehlhorn, 1980; Vitter, 1999; Lai and Wood, 1993; Andersson, 1999) due to its simple construction principles, particularly when a uniform distribution is assumed over all items. Thus, implying that the search set is partitioned to subsets with an equal number of items. For example, Vitter (1999) discussed the use of weight-balanced trees in relation to a variety of on-line data structures for external storage devices in computer applications. Lai and Wood (1993) proposed a top-down restructuring pass to rebalance the tree. Andersson (1999) showed that in order to achieve efficient maintenance of a balanced binary search tree, no shape restriction other than a logarithmic height is required. The obtained class of trees may be maintained at a logarithmic amortized cost with no balance information stored in the nodes.

A large body of literature addresses the problem of optimal testing procedures. The majority of these papers consider binary tests (i.e., implying that a single tester is available at any time point) and focus on modifications and extensions to the Huffman (1952) and the Hu and Tucker (1971) search algorithms. Abrahams (1994) analyzed the performance of a binary Huffman and the Hu-Tucker search algorithms for a simultaneous parallel search. She showed how to set the number of parallel searches to guarantee expected search lengths shorter than some desired constant. Abrahams (1994) also proposed the examination of other aspects of tree search problems, as presented in this paper. Yeung (1991) considered binary-test problems for an ordered set, where ordering is with respect to the probability distribution of components to be the MF. He has shown that if the ordered distributions are ascending or descending, the expected length of an optimal alphabetic code is the same as that of the Huffman code for the unordered distribution. We use this phenomenon when comparing the upper bound for the WBT algorithm with the optimal Hu-Tucker algorithm. Lipman and Abrahams (1995) extended the problem of designing a sequence of optimal binary tests (for identification of a single faulty component) to account for partially ordered components. They solved the problem by reducing it to a series of alphabetic (linearly constrained) minimization problems. Varshney *et al.* (1982) considered a similar problem within the framework of fault diagnosis of electronic systems. In particular, they addressed the problem of the construction of efficient sequential fault-location experiments, and proposed a near-optimum sequential procedure which is computationally tractable and based on information theory principles. Other sources for WBT search problems, are Ahlswede and Wegner (1987) and Du and Hwang (1993).

In an early paper, Horibe (1977) analyzed the expected number of tests of the binary weight-balanced tree. He provided an upper bound on the expected path length of an alphabetical binary tree, which is constructed by the weight balancing algorithm. This paper generalizes some of his ideas to obtain an upper bound on the expected number of tests for a $Q$-ary balanced tree, which represents a multiple testers procedure.

The use of the $Q$-ary balanced tree for searching a damaged product in a manufacturing batch has been recently proposed in Herer and Raz (2000) in relation to Raz (1991). The authors investigated a production system with

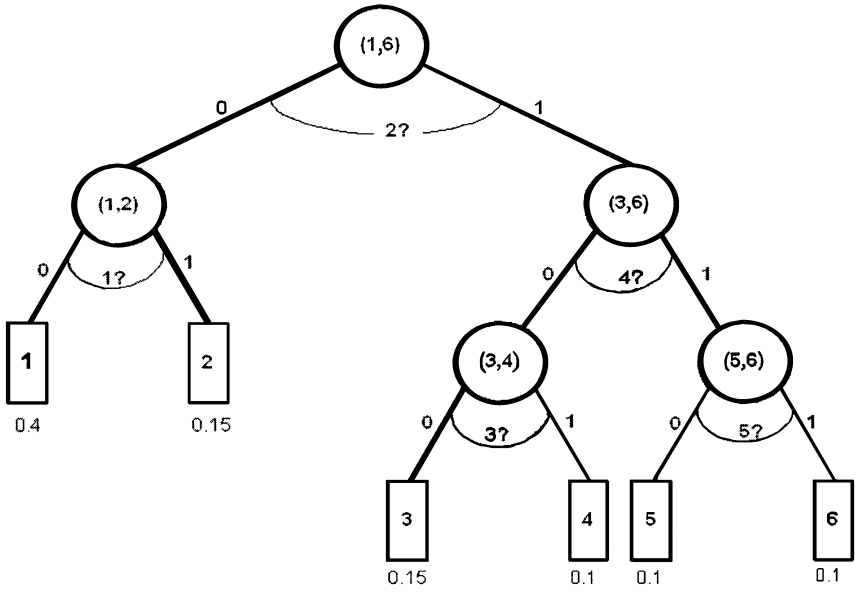


**Fig. 2.** A binary WBT tree for $n = 6$ components.

a constant failure rate. They assumed that after a process failure the remaining units that are produced are nonconforming. Their objective was to find the first nonconforming unit with a minimal expected number of tests. In particular, the authors suggested a parallel inspection procedure with multiple testers that is based on the $Q$-ary WBT, i.e., at any stage divide the search set into $Q$ equiprobable subsets. The authors mentioned the entropy as the ultimate lower bound on the expected number of tests; a fact which is evidently true regardless of the applied testing algorithm. However, they did not provide an upper bound to their suggested method; an upper bound such as the one which is obtained in this paper.

## 3. The $Q$-ary weight-balanced tree and alphabetic codes

A $Q$-ary testing procedure is performed by $m$ testers simultaneously, partitioning the search set into $(Q = m + 1)$ subsets. Each of the testers is allocated to an item (we use the terms item, component and unit interchangeably) in the set and according to the results the next $Q$-ary testing procedure is specified, that is, the next $m$ items to be inspected are selected. The weight-balanced approach seeks to locate the $m$ testers such that they partition the set to $(m + 1)$ equiprobable subsets with respect to the probability of finding the MF unit in that subset. Since the theoretical unit numbers to be inspected are most likely not integers, some simple rounding or ceiling procedure is often applied.

The above testing procedure can be described by a $Q$-ary search tree, where $Q = m + 1$ (Horibe, 1977; Herer and Raz, 2000). The $Q$-ary search tree is a graphic representation of successive division of a set $\{1, \ldots, n\}$ into $Q$ subsets, each resulting subset consisting of consecutive integers. Figure 2 presents, for example, a binary WBT search tree for a single tester ($m = 1$) and six components ($n = 6$). The six terminal nodes (leaves) are labeled by the component numbers. The probabilities of the components to be the MF, $P_i$, $i = 1, \ldots, 6$, are known in advance and shown at the bottom of the terminal nodes. Each internal node represents a subsystem and is labeled by its associated components. The probability that a subsystem includes the MF component is equal to the sum of probabilities of its components that are labeled in the node. The tested component which divides the subsystem in each stage is shown between each pair of arcs (with a question mark). The arcs are labeled by the test outcome, where a one denotes a correct signal and a zero denotes a faulty signal. Based on the tests outcome, the relevant subsystem (including the whole system in the first step) is partitioned in the next step into two sub-subsystems by the following procedure: components are grouped into the first sub-subsystem until its accumulated probability is equal to or larger than half of the probability which is associated with the original undivided subsystem. The resulting subsystems of components at each tree level are labeled by the descendant nodes. The first test is performed on the output of component 2, since the subsystem containing components 1 and 2 is the smallest one with an accumulated probability larger than 0.5. Then, for example, if the result is a one, a second test is performed on the output of component 4, since the subsystem containing components 3 and 4 has an accumulated probability of 0.25. If the result of the second test is a zero, a third test is performed on the output of component 3. If the result of the third

test is a zero, the faulty component is component 3. If the result of the third test is a one, the faulty component is component 4. The expected number of tests in this example is, thus, $L = 2\,(P_1 + P_2) + 3\,(P_3 + P_4 + P_5 + P_6) = 2 \times 0.55 + 3 \times 0.45 = 2.45$.

The relation between the design of a testing procedure and the design of prefix-free codes has long been established (see, for example, Horibe (1977) and for recent publications see, Ben-Gal and Levitin (2001) and also Ben-Gal *et al.* (2002)).

A *code* for a random variable (random source) $X$ is a mapping from the range of $X$ to a set of finite length strings of symbols from a $Q$-ary alphabet. A *prefix-free code* $C$ is a code such that no codeword $w_1 \in C$ is a prefix of another codeword $w_2 \in C$. For example, for a 3-ary alphabet, $C_1 = \{02, 01, 102, 101, 1101\}$ is a prefix-free code, whereas $C_2 = \{02, 01, 100, 121, 1002\}$ is not a prefix-free code, since the third codeword is a prefix of the fifth codeword. Accordingly, a sequence of prefix-free codewords can be decoded instantaneously without reference to future codewords, since the end of a codeword is immediately recognizable. This is the reason why these codes are sometimes also called *instantaneous codes* or *self-punctuating codes*.

Table 1 presents the equivalent prefix-free code to the testing example given in Fig. 2. A codeword, which is represented by a path from the root to a leaf, is associated with the test outcomes related to each component. Thus, an analogy can be established between codewords and testing vectors of components, where elements of the testing vector indicate the series of subsets that should be tested until the component is identified as MF. Accordingly, the expected number of tests is identical to the expected code length and is equal in this case to 2.45.

Shannon (1948) has shown that the entropy function provides the (ultimate) theoretical lower bound on the expected code length and, thus, a lower bound on the expected number of tests. The entropy lower bound is obtainable if and only if the division of the search set to equiprobable subsets is perfect; such a condition is guaranteed for continuous search spaces or for special discrete search sets. If this condition does not hold (such as in the example presented above, where the search space is a general set of discrete components) the Huffman (1952) coding provides the optimal procedure, which is constructible and known to be less than or equal to the entropy plus one test. However, the Huffman coding procedure assumes that the search set is unordered, thus, at any stage, any partition to subsets of components can be tested. When this assumption does not hold, and particularly, when a linear-order constraint is assumed on the set of components, the Hu and Tucker (1971) algorithm provides the optimal procedure, which is approximately the entropy lower bound plus two tests (a refined result can be found in Ahlswede and Wegner (1987) and Abrahams (1994)). In this paper we compare our results to the above optimal testing procedures, the Huffman and the Hu-Tucker algorithms, however, we do not describe them, since they are well addressed in the literature.

**Table 1.** The equivalent prefix-free code

| *Unit* | $P_i$ | *Codeword* | *Length* |
|---|---|---|---|
| 1 | 0.40 | 00 | 2 |
| 2 | 0.15 | 01 | 2 |
| 3 | 0.15 | 100 | 3 |
| 4 | 0.10 | 101 | 3 |
| 5 | 0.10 | 110 | 3 |
| 6 | 0.10 | 111 | 3 |

The fact that the relatively good performance of the weight-balanced algorithm in certain binary codes (known also as *Fano coding*) does not guarantee good performance for $Q$-ary searches is known in coding related literature, but sometimes overlooked in testing related papers. Evidently, the WBT greedy algorithm is far less efficient than the optimal Huffman and the Hu-Tucker procedures, since, at any stage, the WBT divides the search set without considering the next divisions of the resulting subsets. Moreover, in the $Q$-ary WBT algorithm the resulting subsets at each stage might contain less components than testers, thus, "contributing" to the inefficiency of the algorithm. In Section 5, we analyze the WBT algorithm with multiple testers numerically and compare it to the above optimal procedures and their upper bounds. As expected, in most cases the WBT algorithm is less efficient than those procedures. It achieves the Huffman coding's performance if and only if the partition of the search set at any stage will maintain the equiprobability of the resulting subsets until the last stage.

In the following we derive an upper bound for the WBT algorithm with multiple testers for any given distribution for components failure.

## 4. An upper bound on the WBT algorithm with multiple testers

### 4.1. *Notation and definitions*

The search procedure is modeled by constructing an alphabetical $Q$-ary search tree. In the $Q$-ary tree there are $n$ terminal nodes (leaves) and $S$ internal nodes. A terminal node in the tree is a node of degree 1 having no descendent nodes, only a parent node. An internal node has a degree $2 < d \leq Q + 1$, thus, having a parent node and up to $Q$ descendent nodes. The following notation is used:

$m$ = The number of testers available to be functioned simultaneously, partitioning the set $\{i, \ldots, j\}$ to $(m + 1)$ search sets, unless $j - i + 1 < m + 1$.

$Q$ = The alphabet size of the search tree. Correspondingly, $Q = m + 1$.

$T(Q, n)$ = A $Q$-ary WBT tree with $n$ leaves. The search tree represents the testing procedure of $n$ components or equivalently a $Q$-ary alphabetic code containing $n$ codewords.

$(i, j)$ = An *internal node* in the tree, $1 \le i \le j \le n$ corresponding to the subsequence $\{i, i+1, \ldots, j\}$. The subtree whose root is this internal node has terminal nodes $i, i+1, \ldots, j$.

$(i, i)$ = An *terminal node* (a leaf) in the tree, $(i, i) \triangleq i$. Each terminal node represents a component that can be identified by a $Q$-ary codeword representing the outcomes of the tests.

$I$ = The set of all internal nodes, $(i, j) \in I, i \ne j$. $S$ is the cardinality of $I$, i.e., $|I| = S$, where $S \le S_{\max}$, the maximum (theoretical) number of nodes in the WBT tree.

$P_i$ = The probability associated with a terminal node, or equivalently, the probability of the corresponding $i$th component to be the MF, $i = 1, \ldots, n$, $\sum_{i=1}^{n} P_i = 1$, where $P_0 \triangleq 0$. For some of the following observations, we will assume that the probability mass function (pmf) $P_i$ is monotonically decreasing in $i$, i.e., $P_i \ge P_j$ for all $i < j$. However, in general, this condition is not assumed.

$P(i, j)$ = Sum of probability terms associated with an internal node, $P(i, j) = P_i + P_{i+1} + \cdots + P_j$.

$l_i$ = The depth (level) of node $i$, defined as the number of nodes from node $i$ (including) to the tree root (not including). The depth of the node corresponds to the number of tests required by the search tree to identify the $i$th unit, if it is the MF.

$k_{ij}^*(r)$ = The theoretical unit number to be inspected by the $r$th tester, $r = 1, \ldots, m$, $i \le k_{ij}^*(r) < j$, where the $m$ testers partition the set $\{i, i+1, \ldots, j\}$ according to the weight-balanced algorithm. Thus, $P(i, k_{ij}^*(r))/P(i, j) = \frac{r}{(m+1)}$ and $P(k_{ij}^*(r) + 1, j)/P(i, j) = (m - r + 1)/(m + 1)$. In general, $k_{ij}^*(r)$ is not an integer and a ceiling procedure is performed to obtain an integer value. The following observations can be easily modified to account for a floor procedure instead of the ceiling. We define $k_{ij}^*(0) \triangleq i - 1$, $k_{ij}^*(m+1) \triangleq j$.

$\hat{k}_{ij}(r)$ = The actual unit number (an integer) being inspected by the $r$th tester, where $\hat{k}_{ij}(r) \in \{i, i+1 \ldots, j\}$. In the case where the number of units in the subset is less than or equal to the number of testers, i.e., $j - i + 1 \le m$, each unit is inspected by a tester. We omit the subscripts when discussing the entire set, i.e., $\hat{k}_{ij}(r) = \hat{k}(r)$, if $i = 1, j = n$.

### 4.2. *Observations*

The expected number of tests to identify the MF in the set by a given $Q$-ary search tree with $n$ leaves, $T(Q, n)$, is:

$$L(T(Q, n)) = \sum_{i=1}^{n} P_i l_i = \sum_{(i,j)\in I} P(i, j).$$

The equality results from the associative property of the summation of probability terms.

The $Q$-ary entropy of a set of discrete probabilities, which is denoted by $\{P_i\} = P_1, P_2, \ldots, P_n \;\; n \ge 2$, is defined as:

$$H_Q(\{P_i\}) = -\sum_{i=1}^{n} P_i \log_Q P_i,$$

where the log is taken to the base $Q$. We omit the subscript when discussing general properties of entropy. Lemma 1 follows directly from the chain rule of entropy (see, for example, Horibe (1977)).

**Lemma 1.**

$$\begin{aligned} &H(\{P_i\}) \\ &= H(P(1, \hat{k}(1)), P(\hat{k}(1) + 1, \hat{k}(2)), \ldots, P(\hat{k}(m) + 1, n)) \\ &\quad + P(1, \hat{k}(1)) H\left(\frac{1}{P(1, k(1))}, \ldots, \frac{P_{\hat{k}(1)}}{P(1, \hat{k}(1))}\right) + P(\hat{k}(1) \\ &\quad + 1, \hat{k}(2)) H\left(\frac{P_{\hat{k}(1)+1}}{P(\hat{k}(1) + 1, \hat{k}(2))}, \ldots, \frac{P_{\hat{k}(2)}}{P(\hat{k}(1) + 1, \hat{k}(2))}\right) \cdots \\ &\quad + P(\hat{k}(m) + 1, n) H\left(\frac{P_{\hat{k}(m)+1}}{P(\hat{k}(m) + 1, n)}, \ldots, \frac{P_n}{P(\hat{k}(m) + 1, n)}\right), \end{aligned}$$

*and in general*,

$$\begin{aligned} &H(\{P_i\}) = \sum_{(i,j)\in I} P(i, j) \times H \\ &\left(\frac{P(i, \hat{k}_{ij}(1))}{P(i, j)}, \frac{P(\hat{k}_{ij}(1) + 1, \hat{k}_{ij}(2))}{P(i, j)}, \ldots, \frac{P(\hat{k}_{ij}(m) + 1, j)}{P(i, j)}\right). \end{aligned}$$

Notice that in Fig. 2, for example, $H(\{P_i\}) = 2.346$. Lemma 2 follows from Lemma 1 and the definition of $L(T(Q, n))$.

**Lemma 2.**

$$\begin{aligned} &L(T(Q, n)) - H_Q(\{P_i\}) = \sum_{(i,j)\in I} P(i, j) \Bigg\{ 1 - H_Q \\ &\left(\frac{P(i, \hat{k}_{ij}(1))}{P(i, j)}, \frac{P(\hat{k}_{ij}(1) + 1, \hat{k}_{ij}(2))}{P(i, j)}, \ldots, \frac{P(\hat{k}_{ij}(m) + 1, j)}{P(i, j)}\right) \Bigg\}. \end{aligned}$$

**Lemma 3.** *For any set of discrete probabilities* $\{P_i\}$:

$$H_Q(\{P_i\}) \ge (1 - P_{\max}) \log_Q 4,$$

*where* $P_{\max} \triangleq \max_i \{P_i\}$.

**Proof.** The proof makes use of the binary entropy function, $H_2(x) \stackrel{\Delta}{=} -x \log_2 x - (1-x)\log_2(1-x)$, which is convex in $0 \le x \le 1$. Note that for $x \ge 0.5$, $H_2(x) \ge 2(1-x)$, where equality is achieved at $x = 0.5$ and at $x = 1$. Then, recall that $H_Q(\{P_i\}) \ge H_Q(P_{\max}, 1 - P_{\max})$. Thus, $H_2(\{P_i\}) \ge 2(1 - P_{\max})$ for $P_{\max} \ge 0.5$.

Now, note that for $x \le 0.5$, $-\log_2 x \ge 2(1-x)$, where equality is achieved at $x = 0.5$. Then, recall that $H_Q(\{P_i\}) = -\sum_{i=1}^{n} P_i \log_Q P_i \ge -\sum_{i=1}^{n} P_i \log_Q P_{\max} = -\log_Q P_{\max}$. Thus, $H_2(\{P_i\}) \ge 2(1 - P_{\max})$ not only for $P_{\max} \ge 0.5$ but also for $P_{\max} \le 0.5$. Finally, multiply both sides of the equation by $\log_Q 2$ to obtain the lemma. ■

Lemma 4 follows from Lemmas 2 and 3.

**Lemma 4.**

$$L(T(Q,n)) - H_Q(\{P_i\}) \le \sum_{(i,j)\varepsilon I} P(i,j)\Delta_{ij},$$

*where*

$$\Delta_{ij} = 1 - \left(1 - \max\left\{\frac{P(i,\hat{k}_{ij}(1))}{P(i,j)}, \ldots, \frac{P(\hat{k}_{ij}(m)+1,j)}{P(i,j)}\right\}\right)\log_Q 4.$$

The value of $\Delta_{ij}$ depends on the weight-balanced algorithm, the search probabilities and on the ceiling procedure which is applied to obtain $\hat{k}_{ij}(r), r = 1, \ldots, m$ and considered next.

**Lemma 5.** *Let the actual inspected units to be obtained by a ceiling procedure, thus,* $\hat{k}_{ij}(r) = \lceil k^*_{ij}(r) \rceil, r = 1, \ldots, m$. *Then*:

$$P(\hat{k}_{ij}(r-1)+1, \hat{k}_{ij}(r)) \le \frac{P(i,j)}{m+1} + P_{\hat{k}_{ij}(r)} \le \frac{P(i,j)}{m+1} + P_{\max}.$$

**Proof.** Note that:

$$\frac{r}{m+1} \le \frac{P(i,\hat{k}_{ij}(r))}{P(i,j)} \le \frac{r}{m+1} + P_{\hat{k}_{ij}(r)}/P(i,j),$$

since, by definition, $P(i,\hat{k}_{ij}(r))/P(i,j) \ge r/(m+1)$ and $P(i,\hat{k}_{ij}(r)-1)/P(i,j) = P(i,\hat{k}_{ij}(r))/P(i,j) - P_{\hat{k}_{ij}(r)}/P(i,j) \le r/(m+1)$. It thus follows that:

$$\begin{aligned} &P(\hat{k}_{ij}(r-1)+1, \hat{k}_{ij}(r))/P(i,j) \\ &\quad = P(i,\hat{k}_{ij}(r))/P(i,j) - P(i,\hat{k}_{ij}(r-1))/P(i,j) \\ &\quad \le \left(\frac{r}{m+1} + P_{\hat{k}_{ij}(r)}/P(i,j)\right) - \left(\frac{r-1}{m+1}\right) \\ &\quad = \frac{1}{m+1} + P_{\hat{k}_{ij}(r)}/P(i,j), \end{aligned}$$

where the first equality follows from the definition of $\hat{k}_{ij}(r)$ and the inequality is obtained by replacing the first probability sum by its maximum value and the second probability sum by its minimum value. The second inequality in Lemma 5 follows from the definition of $P_{\max}$, which is equal to $P_1$ if the probability is monotonically non-increasing. ■

Observation 1 follows from Lemmas 4 and 5.

**Observation 1.** *An upper bound on the expected number of tests for a given WBT tree with respect to the entropy is given by*:

$$\begin{aligned} L(T(Q,n)) - H_Q(\{P_i\}) &\le \sum_{(ij)\epsilon I} P(i,j)\Delta_{ij} \le \sum_{(ij)\epsilon I} P(i,j) \\ &\quad \times \left(1 - \left(\frac{m}{m+1}\right)\log_Q 4\right) + \log_Q 4 \sum_{(ij)\epsilon I} P_{\max}. \end{aligned}$$

**Proof.** Let us apply Lemma 5 to Lemma 4:

$$\begin{aligned} \sum_{(i,j)\varepsilon I} P(i,j)\Delta_{ij} &= \sum_{(i,j)\varepsilon I} P(i,j)\left[1 - \left(1 - \max\left\{\frac{P(i,\hat{k}_{ij}(1))}{P(i,j)}, \ldots, \frac{P(\hat{k}_{ij}(m)+1,j)}{P(i,j)}\right\}\right)\log_Q 4\right], \\ &\le \sum_{(i,j)\varepsilon I} P(i,j) - \log_Q 4 \sum_{(i,j)\varepsilon I} P(i,j)\left(1 - \frac{1}{P(i,j)}\left(\frac{P(i,j)}{m+1} + P_{\max}\right)\right), \\ &= \sum_{(ij)\epsilon I} P(i,j)\left(1 - \left(\frac{m}{m+1}\right)\log_Q 4\right) + \log_Q 4 \sum_{(ij)\epsilon I} P_{\max}, \end{aligned}$$

where the inequality follows from Lemma 5. ■

Although the construction of the WBT tree is compositionally tractable (of the order $O(n^2)$, e.g., see Ahlswede and Wegner (1987)) the above upper bound is not a practical one since it depends explicitly on the tree structure. Thus, it requires one to construct the WBT tree, obtain the set of internal nodes $P(i,j), (i,j) \in I$ and compute the expected number of tests for any possible vector of probability terms $\{P_i\}$. It is therefore of interest to obtain an upper bound that does not require such construction explicitly.

In order to obtain a closed-form upper bound on the expected number of tests of the WBT algorithm regardless

of the structure of the tree, one needs to derive bounds for both the number of nodes in the tree, $\sum 1_{(i,j)\epsilon I} = |I| = S$, and their respective sum of probabilities $P(i, j)$. Note, however, that in general as the number of nodes in the tree increases the respective sum of probabilities decreases and *vice versa*. Next, we obtain such a closed-form upper bound by considering the maximum number of nodes that are theoretically feasible for a weight-balanced tree, while associating each level in the tree with the highest theoretic sum of probabilities.

**Lemma 6.** *The maximum number of internal nodes in a* $(m+1)$*-ary weight-balanced tree with n terminal nodes* (*leaves*) *is*:

$$S_{\max} \triangleq \max_{\text{all } (m+1)\text{-ary trees with } n \text{ leaves}} S \leq \left\lceil \frac{n(m+1)}{2m} - 1 \right\rceil.$$

**Proof.** In order to maximize the number of nodes in a $Q$-ary tree with $n$ leaves, $T(Q, n)$, one needs to maximize the number of internal nodes in the tree since the number of terminal nodes is fixed to $n$. The minimum number of descendent leaves in an internal node is two (a node with one descendant leaf is superfluous). Accordingly, one can construct a tree with maximum internal nodes by ensuring that at each level of the tree there will be exactly $(Q-1)$ internal nodes with two descendent leaves and a $Q$th node, which is associated with all the remaining leaves (the last node in each level if the probability is monotonically decreasing), as exemplified in Fig. 3.

The number of levels, or alternatively, the maximum depth of such tree is:

$$\left\lfloor \frac{n}{2(Q-1)} \right\rfloor,$$

where at each level there are $Q$ internal nodes. Thus, the number of internal nodes cannot exceed:

$$\left\lceil \frac{nQ}{2(Q-1)} - 1 \right\rceil.$$

Recalling that $Q = m + 1$, the lemma follows. In fact:

$$\left\lfloor \frac{n}{2m} \right\rfloor (m+1) - 1 \leq S_{\max} \leq \left\lfloor \frac{n}{2m} \right\rfloor (m+1) + N \bmod (2m) - m,$$

as seen in the Appendix. ■

**Lemma 7.** *A maximum value for the sum of probability terms of the internal nodes in the tree,* $\sum_{(ij)\epsilon I} P(i, j)$*, is given by*:

$$\sum_{(ij)\epsilon I} P(i,j) < \log_{m+1}(1 + mS_{\max}) = \log_{m+1}\left(1 + m\left\lceil \frac{n(m+1)}{2m} - 1 \right\rceil\right).$$

**Proof.** We consider a weight balanced tree, $T(Q, n)$, which is entirely complete and balanced, thus, in each level (depth) $l = 1, 2, \ldots$ in the tree there are $Q^l$ nodes, each of which has a probability of $Q^{-l}$, as exemplified in Fig. 4. In such a tree, each level contributes one, which is the maximal value, to the sum of probability terms that equals $\hat{l}$, where $\hat{l}$ is the maximal depth in the tree. If the number of internal nodes in the tree is $S$, then:

$$S = 1 + Q + Q^2 + \cdots + Q^{\hat{l}-1} = \frac{1 - Q^{\hat{l}}}{1 - Q}.$$

For $Q = m + 1$, it follows that $\hat{l} = \log_{m+1}(1 + mS)$. The lemma is obtained by substituting $S$, the actual number

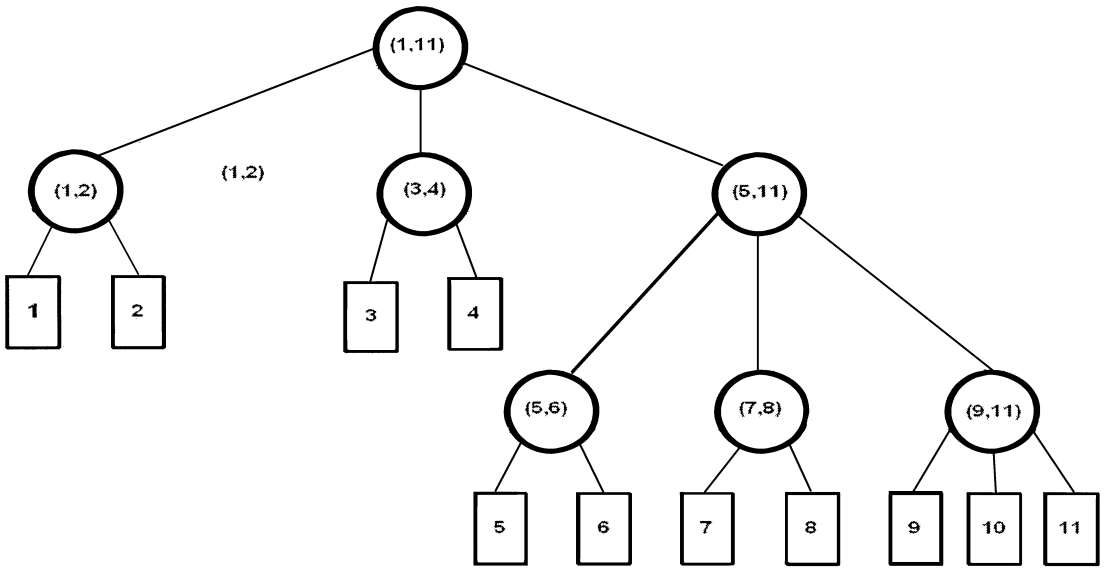


**Fig. 3.** A $Q$-ary tree with a maximum number of internal nodes.

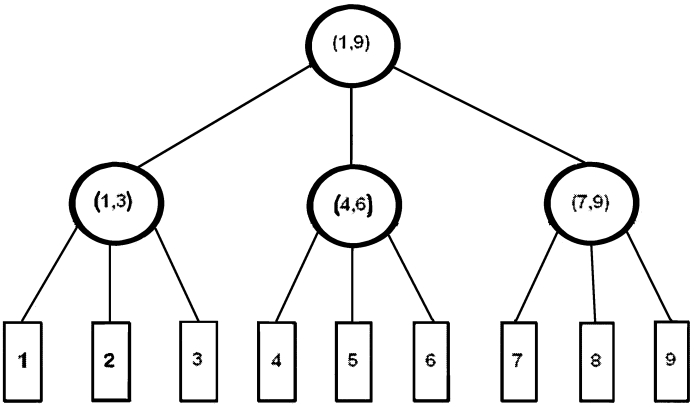


**Fig. 4.** A complete and balanced tree with $m = 2, n = 9, \hat{l} = 2, S = 4, S_{\max} = 6$.

of internal nodes in such a tree, with the maximum number of nodes $S_{\max}$, where $S_{\max} \geq S$. Recall that $S_{\max}$ is obtained for the non-complete tree in Lemma 6, which contains the maximum number of internal nodes. ■

**Observation 2.** *A closed-form upper bound on the expected number of tests for a given WBT, T(Q,n), is given by*:

$$L(T(Q,n)) - H_Q(\{P_i\}) \leq \sum_{(i,j)\varepsilon I} P(i,j)\Delta_{ij}$$

$$< \log_{m+1}\left(1 + m\left\lceil \frac{n(m+1)}{2m} - 1 \right\rceil\right)\left(1 - \left(\frac{m}{m+1}\right)\log_Q 4\right) + \log_Q 4\left\lceil \frac{n(m+1)}{2m} - 1 \right\rceil P_{\max}.$$

**Proof.** The inequalities follow directly from Observation 1 and Lemmas 6 and 7. ■

**Observation 3.** *Note that for large n and m values, the following (simplified) version of the above upper bound might be tighter*:

$$L(T(Q,n)) - H_Q(\{P_i\}) \leq \log_{m+1}\left(1 + m\left\lceil \frac{n(m+1)}{2m} - 1 \right\rceil\right) - (1 - P_{\max})\log_Q 4.$$

**Proof.**

$$L(T(Q,n)) = \sum_{(i,j)\varepsilon I} P(i,j) \leq \log_{m+1}\left(1 + m\left\lceil \frac{n(m+1)}{2m} - 1 \right\rceil\right),$$

as indicated in Lemma 7 and $H_Q(\{P_i\}) \geq (1 - P_{\max})\log_Q 4$, as indicated in Lemma 3.

■

## 5. Example: The geometric distribution

Let us now consider the case where the probability of a unit to be the MF follows the geometric distribution. However, note that all the search procedures (and their bounds) that we analyze in this example are constructed for a general (discrete) probability distribution and make no particular use of the geometric distribution assumptions. Presumably, some of these procedures could be refined and improved, if limited to the geometric distribution case. For example, the best search solution for a single-tester search in the geometric distribution case can be found in Gallager and Voorhis (1975). At the end of this section we also analyze some cases that are not based on the geometric distribution.

Following He *et al*. (1996) and Herer and Raz (2000), we consider a production process with a constant failure rate, where the probability that the *i*th unit, $i = 1, \ldots, n$, is malfunctioning is given by geometric distribution, i.e.:

$$P_i = \frac{(1-q)q^{i-1}}{1-q^n},$$

where $q$ is the probability parameter of the geometric distribution and $(1 - q)$ is the failure rate.

Applying the above observations regarding the $Q$-ary WBT algorithm with a ceiling procedure yields the following results.

First, recall that in a search set with $n$ components, the geometric accumulated distribution of the first $j$ components is $\sum_{i=1}^{j} P_i = P(1,j) = (1 - q^j)/(1 - q^n)$. According to the definition of $k^*_{ij}(r)$ we require that:

$$\frac{1 - q^{k^*_{ij}(r)-i+1}}{1 - q^{j-i+1}} = \frac{r}{m+1},$$

and solve it for $k^*_{ij}(r)$ to obtain:

$$k^*_{ij}(r) = (i-1) + \log_q \frac{(m+1) + r(q^{j-i+1} - 1)}{(m+1)}.$$

Then, following Observation 1, and recalling that the geometric distribution is monotonically decreasing in the unit index, where $P_{\max} = P_1$, we obtain the upper bound:

$$L(T(Q,n)) - H_Q(\{P_i\}) \leq \sum_{(ij)\epsilon I} P(i,j)\left(1 - \left(\frac{m}{m+1}\right)\log_Q 4\right) + \log_Q 4 \sum_{(ij)\epsilon I} \frac{1-q}{1-q^n}.$$

**Observation 4.** *A closed-form upper bound for the Q-ary WBT algorithm with a ceiling procedure, where components failure follow a geometric distribution with parameter q*, is:

$$L(T(Q,n)) - H_Q(\{P_i\}) < \log_{m+1}\left(1 + m\left\lceil \frac{n(m+1)}{2m} - 1 \right\rceil\right) \times \left(1 - \left(\frac{m}{m+1}\right)\log_Q 4\right) + \log_Q 4\left\lceil \frac{n(m+1)}{2m} - 1 \right\rceil \frac{1-q}{1-q^n}.$$

**Proof.** Substitute $P_{\max}$ with $P_1 = (1 - q)/(1 - q^n)$ in Observation 2. ■

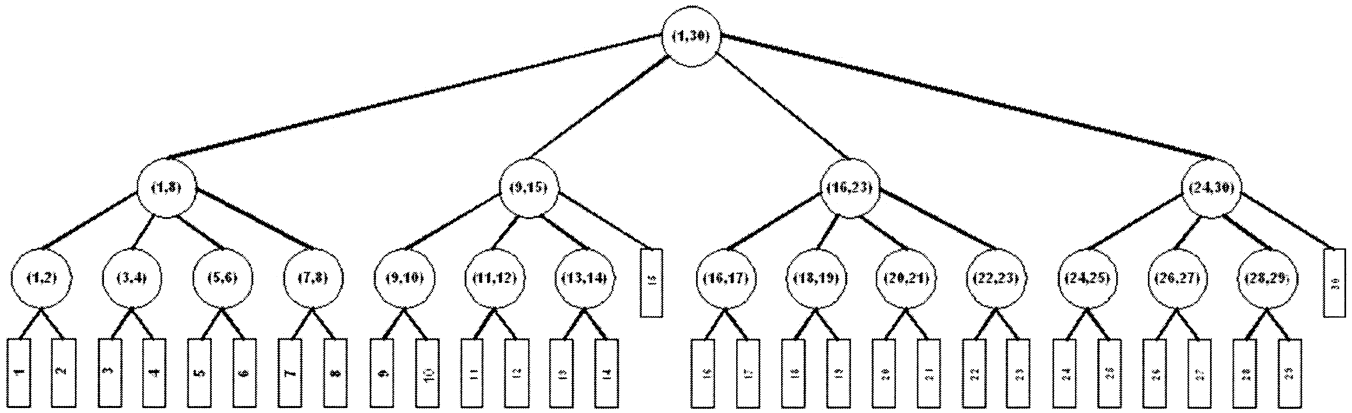


**Fig. 5.** A WBT tree for the geometric distribution $m = 3$, $n = 30$, $q = 0.999$.

Figure 5 presents an example for an WBT tree with $m = 3$ testers and $n = 30$ components, where the probability of a component to be MF follows a geometric distribution with parameter $q = 0.999$, resulting from a constant failure rate of 0.001. The geometric distribution is monotonically decreasing in the component index, thus, component number 1 has the highest probability of being the MF, $P_1 = 0.0338$, and component number 30 has the lowest probability of being the MF, $P_{30} = 0.0328$. Note that the tree represents a testing strategy which is far from being optimal; the equiprobable division at the first two levels results with two-leaves branches in the last level, instead of four leaves. It is evident that an efficient testing policy should assign shorter testing branches to less reliable components and *vice versa*, since this will result in a lower expected number of tests. However, in this example it is seen that the WBT fails to do so since the greedy partition of the search set in the upper levels creates non-optimal situations in the lower levels. In fact, component number 30, which has the lowest probability of being the MF, requires two tests, whereas component number 1, which has the highest probability of being the MF, requires three tests. The entropy that represents the ultimate lower bound on the expected number of tests (a bound which is unattainable in this case) is equal to 2.45 tests. The Huffman code length that represents the lower upper bound for the discrete search space that can be achieved for such distribution is 2.63 tests. The actual WBT expected number of tests is 2.93 tests. The WBT upper bound that follows from Observation 4 is equal to 3.82 tests, thus, 1.37 tests above the entropy. The rough Hu-Tucker upper bound is, thus, higher in this case than the WBT upper bound and equals 4.45 tests; two tests above the entropy.

Table 2 follows a similar numerical analysis as above. It presents a comparative study of the expected number of tests and its various bounds for the following functions and procedures: (i) the entropy function; (ii) the Huffman coding; (iii) the Hu-Tucker coding (denoted by H-T); (iv) the WBT expected number of tests; (v) the Hu-Tucker upper bound (denoted by H-T UB); and (vi) the WBT upper bound (denoted by WBT UB).

The expected number of tests of the above procedures depends on the combination of three input parameters: (i) the number of components, denoted by $n$; (ii) the number of testers, denoted by $m$; and (iii) the probability parameter of the geometric distribution, denoted by $q$.

The table is partitioned into two parts. Rows 1–28 present various geometric distribution cases, where the Hu-Tucker procedure achieves the performance of the Huffman procedure. Rows 29–42 present some non-geometric distributions, where the Hu-Tucker and the Huffman procedures can result in a different expected number of tests. In all but three cases (rows 34, 37 and 40), the expected number of tests by the WBT algorithm is larger than the expected number of tests by the Hu-Tucker procedure, although in some cases they are close. We now describe some of the main results in the table.

As can be seen by the first five rows (1–5), the expected number of tests increases with $n$: the number of components. Note that for all cases the WBT upper bound is below the rough Hu-Tucker upper bound, which is equal to the entropy + 2. In fact, this is true as long as $n$ is smaller than 445 (for $q = 0.999$ and $m = 5$), as shown in Fig. 6. In these rows, the WBT expected number of tests is 104–113% of the Hu-Tucker expected number of tests.

Rows 6–10 present the effect of the number of testers, $m$, on the expected number of tests for fixed $n = 3000$ and $q = 0.999$. Note that the WBT expected number of tests decreases in the number of testers. The WBT expected number of tests is 106–114% of the Hu-Tucker expected number of tests. The WBT upper bound for such a large $n$ value, is above the Hu-Tucker upper bound.

Rows 11–15 repeat the same study on the effects of $m$ for a smaller search set, where $n = 120$. In these cases, the WBT expected number of tests is 100–112% of the Hu-Tucker expected number of tests. Note, however, that the WBT upper bound, for $m \geq 2$ is less than the Hu-Tucker upper

**Table 2.** A comparative study of the expected number of tests

| *Number* | *M* | *N* | *q* | *Entropy* | *Huffman* | *H-T* | *WBT* | *H-T UB* | *WBT UB* |
|---|---|---|---|---|---|---|---|---|---|
| Geometric distribution cases | | | | | | | | | |
| 1 | 5 | 60 | 0.999 | 2.285 | 2.476 | 2.476 | 2.800 | 4.285 | 3.775 |
| 2 | 5 | 80 | 0.999 | 2.446 | 2.654 | 2.654 | 3.000 | 4.446 | 4.002 |
| 3 | 5 | 100 | 0.999 | 2.570 | 2.761 | 2.761 | 3.000 | 4.570 | 4.178 |
| 4 | 5 | 120 | 0.999 | 2.672 | 2.834 | 2.834 | 3.000 | 4.672 | 4.322 |
| 5 | 5 | 140 | 0.999 | 2.758 | 2.886 | 2.886 | 3.000 | 4.758 | 4.445 |
| 6 | 30 | 3000 | 0.999 | 2.242 | 2.378 | 2.378 | 2.729 | 4.242 | 4.807 |
| 7 | 40 | 3000 | 0.999 | 2.073 | 2.150 | 2.150 | 2.393 | 4.073 | 4.565 |
| 8 | 50 | 3000 | 0.999 | 1.958 | 2.026 | 2.026 | 2.236 | 3.958 | 4.397 |
| 9 | 60 | 3000 | 0.999 | 1.873 | 1.987 | 1.987 | 2.149 | 3.873 | 4.271 |
| 10 | 70 | 3000 | 0.999 | 1.806 | 1.970 | 1.970 | 2.096 | 3.806 | 4.172 |
| 11 | 1 | 120 | 0.999 | 6.906 | 6.930 | 6.930 | 6.934 | 8.906 | 9.010 |
| 12 | 2 | 120 | 0.999 | 4.357 | 4.477 | 4.477 | 4.650 | 6.357 | 6.100 |
| 13 | 4 | 120 | 0.999 | 2.974 | 2.991 | 2.991 | 3.000 | 4.974 | 4.638 |
| 14 | 6 | 120 | 0.999 | 2.460 | 2.679 | 2.679 | 3.000 | 4.460 | 4.101 |
| 15 | 8 | 120 | 0.999 | 2.179 | 2.353 | 2.353 | 2.650 | 4.179 | 3.809 |
| 16 | 5 | 140 | 0.950 | 2.212 | 2.268 | 2.268 | 2.487 | 4.212 | 6.621 |
| 17 | 5 | 140 | 0.965 | 2.396 | 2.453 | 2.453 | 2.653 | 4.396 | 5.855 |
| 18 | 5 | 140 | 0.980 | 2.603 | 2.697 | 2.697 | 3.023 | 4.603 | 5.163 |
| 19 | 5 | 140 | 0.995 | 2.747 | 2.856 | 2.856 | 3.000 | 4.747 | 4.579 |
| 20 | 5 | 140 | 0.999 | 2.758 | 2.886 | 2.886 | 3.000 | 4.758 | 4.445 |
| 21 | 1 | 50 | 0.999 | 5.644 | 5.715 | 5.715 | 5.721 | 7.644 | 7.652 |
| 22 | 1 | 50 | 0.99 | 5.629 | 5.668 | 5.668 | 5.707 | 7.629 | 8.110 |
| 23 | 2 | 100 | 0.999 | 4.191 | 4.280 | 4.280 | 4.380 | 6.191 | 5.895 |
| 24 | 2 | 100 | 0.99 | 4.154 | 4.195 | 4.195 | 4.347 | 6.154 | 6.350 |
| 25 | 3 | 150 | 0.999 | 3.614 | 3.753 | 3.753 | 4.000 | 5.614 | 5.352 |
| 26 | 3 | 150 | 0.99 | 3.550 | 3.619 | 3.619 | 3.871 | 5.550 | 5.849 |
| 27 | 4 | 200 | 0.999 | 3.291 | 3.445 | 3.445 | 3.747 | 5.291 | 5.079 |
| 28 | 4 | 200 | 0.99 | 3.197 | 3.243 | 3.243 | 3.439 | 5.197 | 5.630 |
| Non-geometric distribution cases | | | | | | | | | |
| 29 | 5 | 60 | 0.999 | 2.285 | 2.476 | 2.483 | 2.800 | 4.285 | 3.775 |
| 30 | 5 | 80 | 0.999 | 2.446 | 2.654 | 2.662 | 3.000 | 4.446 | 4.002 |
| 31 | 5 | 100 | 0.999 | 2.570 | 2.761 | 2.769 | 3.000 | 4.570 | 4.178 |
| 32 | 5 | 120 | 0.999 | 2.672 | 2.834 | 2.841 | 3.000 | 4.672 | 4.322 |
| 33 | 5 | 140 | 0.999 | 2.758 | 2.886 | 2.892 | 3.000 | 4.758 | 4.445 |
| 34 | 1 | 100 | N/A | 6.644 | 6.720 | 6.720 | 6.720 | 8.644 | 8.624 |
| 35 | 2 | 100 | N/A | 4.192 | 4.290 | 4.290 | 4.440 | 6.192 | 5.849 |
| 36 | 5 | 100 | N/A | 2.570 | 2.770 | 2.770 | 2.990 | 4.570 | 4.155 |
| 37 | 1 | 100 | N/A | 6.562 | 6.627 | 6.627 | 6.627 | 8.562 | 9.196 |
| 38 | 2 | 100 | N/A | 4.140 | 4.193 | 4.193 | 4.280 | 6.140 | 6.105 |
| 39 | 5 | 100 | N/A | 2.539 | 2.693 | 2.693 | 2.893 | 4.539 | 4.274 |
| 40 | 1 | 100 | N/A | 6.562 | 6.627 | 6.720 | 6.720 | 8.562 | 9.196 |
| 41 | 2 | 100 | N/A | 4.140 | 4.193 | 4.260 | 4.440 | 6.140 | 6.105 |
| 42 | 5 | 100 | N/A | 2.539 | 2.693 | 2.767 | 2.887 | 4.539 | 4.072 |

bound. Figure 7 compares the above testing procedures and their bounds for $m = 3, \ldots, 10$, $n = 140$ and $q = 0.999$.

Rows 16–20 show the effect of the probability parameter, $q$, on the expected number of tests. Note that although the WBT upper bound decreases in $q$, the WBT expected number of tests is non-monotonic in $q$. The reason for this is that the WBT expected number of tests is affected by the probability of a component to be a MF, $P_i$, which is itself non-monotonic in $q$. However, the WBT upper bound is affected only by $P_{\max} = P_1$, which decreases monotonically in $q$. Moreover, note that as $q \to 1$, two conflicting effects come into play. On the one hand, $P_i$ decreases; facilitating the WBT algorithm to refine the subsets' sum of probabilities. Whereas, on the other hand, the cardinalities of these subsets differ more and are, thus, less effective for subdivisions in lower tree levels. In these rows, the WBT expected number of tests is 104–109% of the Hu-Tucker expected number of tests. It is not surprising that at some point the WBT upper bound lies below the Hu-Tucker upper bound, since the former decreases in $q$ while the latter increases in $q$.

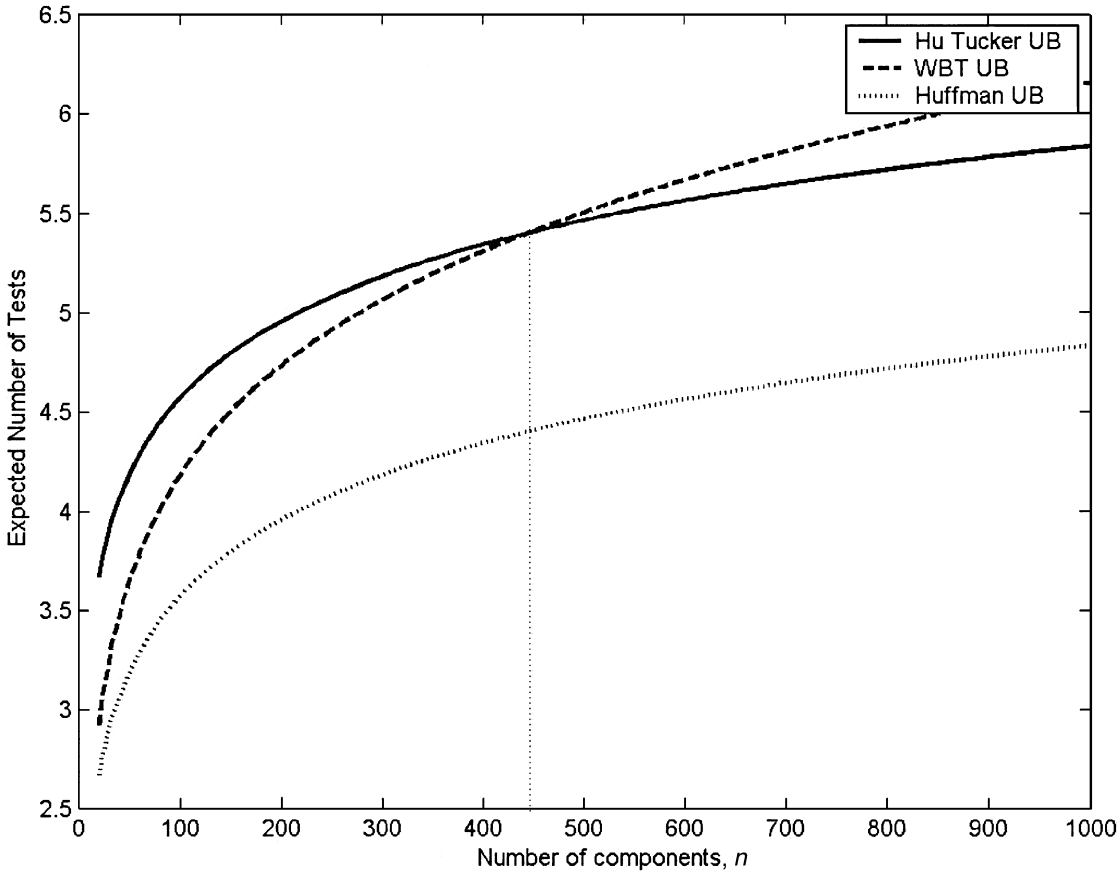


**Fig. 6.** The WBT upper bound as a function of the Hu-Tucker and Huffman upper bounds ($m = 5$, $q = 0.999$).

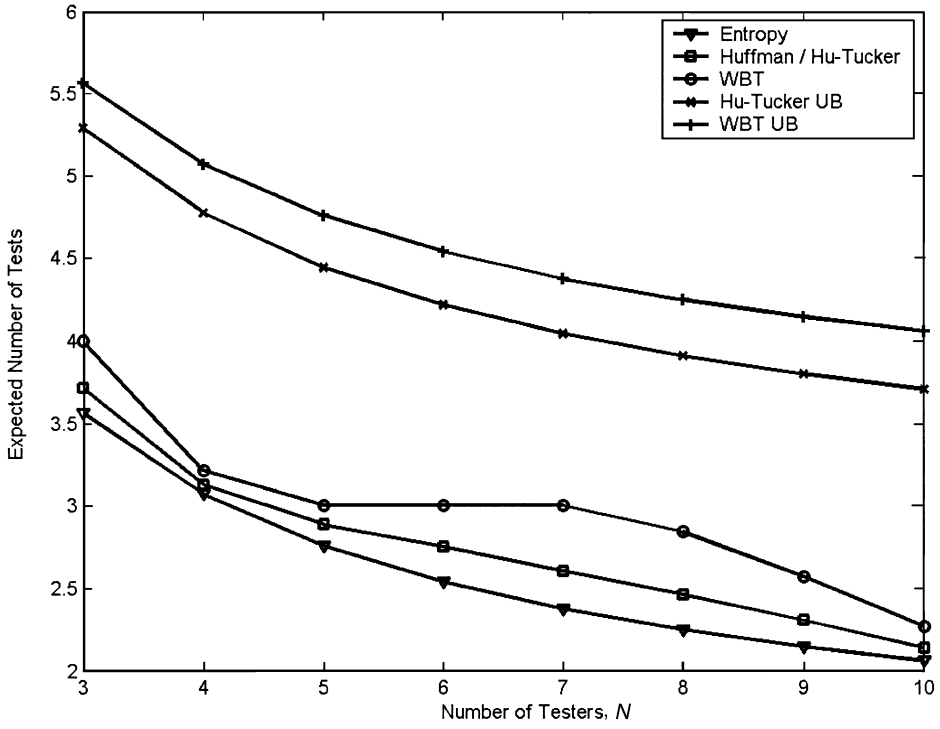


**Fig. 7.** A comparative study on the expected number of tests ($n = 140$, $q = 0.999$).

Rows 21–28 present the combined effects of the various input parameters on the expected number of tests. In particular, the ratio $n/m$ is kept fixed for four different values of $n, m$ and for two values of $q$. Comparing rows 21, 23, 25 and 27 (or rows 22, 24, 26 and 28) one can see that both the expected number of tests and the upper bounds for all the search procedures decrease in $m$ although $n$ is increased proportionally. In these rows, the WBT expected number of tests is 101–109% of the Hu-Tucker expected number of tests. The WBT upper bound is less than the Hu-Tucker upper bound for almost all rows where $q = 0.999$, while getting very close to it in the remaining rows.

The second part of Table 2 includes some cases where the probabilities of units to be the MF are not geometrically distributed. In rows 29–33, the probability values that are generated by the geometric distribution are reordered to produce a highly non-monotonic probability vector. In particular, the new vector has the smallest probability value as its first term, then the largest probability value as its second term, then the second-to-the-smallest value in the third place, then the second-to-the-largest value in the fourth place and so on. As seen, in this case, the Huffman and the Hu-Tucker procedures result in a different expected number of tests.The WBT expected number of tests is 104–113% of the Hu-Tucker expected number of tests. The WBT upper bound is less than the Hu-Tucker upper bound for all cases.

Rows 34–36 are based on a uniform distribution with a fixed $P_i = 0.01$. Rows 37–39 are based on a probability vector where $P_i = 1/150$ for $i = 1, \ldots, 50$ and $P_i = 2/150$ for $i = 51, \ldots, 100$. Note that in all these cases the WBT expected number of tests is very close to the expected number of tests for the Huffman\Hu-Tucker procedures.

Finally, rows 40–42 are based on the same probability vectors of rows 37–39 that are now reordered, such that $P_i = 1/150$ for $i = 1, 3, 5, \ldots, 99$ and $P_i = 2/150$ for $i = 2, 4, 6, \ldots, 100$. Such a reordering imposes a constraint on the Hu-Tucker algorithm, which results in a higher expected number of tests with respect to the Huffman procedure. In these rows, the WBT expected number of tests is within 104% of the Hu-Tucker expected number of tests.

## 6. Summary

This paper studies the performance of the WBT algorithm with multiple testers, and in particular, derives an upper bound on the expected number of tests. The obtained upper bound is applicable for any finite search set with a known probability distribution. It does not require the explicit construction of the testing procedure, but rather the number of testers, the number of units in the set and the relative reliability of the less reliable unit in the set.

This paper also shows the analogy between prefix-free codes and testing procedures, both being represented by a search tree. Such an analogy is particularly important for practitioners and researchers in the areas of quality control and reliability, since it enables them to apply known results from coding theory to these areas (Ben-Gal and Levitin, 2001; Ben-Gal *et al.*, 2002). For example, the use of the Hu-Tucker and the Huffman testing procedures that are known to be optimal for the considered problems. Further research in this area can be performed to refine the bounds of the WBT and other applied testing procedures.

## Acknowledgements

The author would like to acknowledge Dr. Yale Herer and Dr. Tzvi Raz for introducing him to the problem. This research was partially supported by a MAGNET CONSIST Grant, Israel Ministry of Industry and Trade.

## References

Abrahams, J. (1994) Parallelized Huffman and Hu-Tucker searching. *IEEE Transactions on Information Theory*, **40**(2), 508–510.

Ahlswede, R. and Wegner, I. (1987) Search Problems, Wiley, New York, NY.

Andersson, A. (1999) General balanced trees. *Journal of Algorithms*, **30**(1), 1–18.

Ben-Gal, I., Herer, Y. and Raz, T. (2001) Self-correcting inspection sequences under inspection errors. *IIE Transactions on Quality and Reliability*, **34**(6), 529–540.

Ben-Gal, I. and Levitin, L. (2001) An application of information theory and error-correcting codes to fractional factorial experiments. *Journal of Statistical Planing and Inference*, **92**(1/2), 267–282.

Blum, N. and Mehlhorn, K. (1980) On the average number of rebalancing operations in weight-balanced trees. *Theoretical Computer Science*, **11**(3), 303–320.

Du, D.Z. and Hwang, F.K. (1993) *Combinatorial Group Testing and its Applications*, World Scientific, Singapore.

He, Q., Gerchak, Y. and Grosfeld-Nir, A. (1996) Optimal inspection order when process failure rate is constant. *International Journal of Reliability, Quality and Safety Engineering*, **3**(1), 25–41.

Herer, Y.T. and Raz, T. (2000) Optimal parallel inspection for finding the first nonconforming unit in a batch—An information theoretic approach. *Management Science*, **46**(6), 845–857.

Horibe, Y. (1977) An improved bound for weight-balanced tree. *Information and Control*, **34**, 148–151.

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

Huffman, D.A. (1952) A method for the construction of minimum redundancy codes. *Proceedings Institute of Radio Engineering*, **40**, 1098–1101.

Gallager, R.G. and Voorhis, D.C.V. (1975) Optimal source codes for geometrically distributed integer alphabets. *IEEE Transactions on Information Theory*, **21**(2), 228–230.

Lai, T.W. and Wood, D. (1993) A top-down updating algorithm for weight-balanced trees. *International Journal of Foundations of Computer Science*, **4**, 309–324.

Lipman, M.J. and Abrahams, J. (1995) Minimum average cost testing for partially ordered component. *IEEE Transactions on Information Theory*, **41**(1), 287–291.

Raz, T. (1991) Information theoretic measures of inspection. *International Journal of Production Research*, **29**(5), 913–926.

Shannon, C.E. (1948) A mathematical theory of communication. *Bell Systems Technical Journal*, **27**, part I, 379–423; part II, 623–656.

Varshney, P.K., Hartmann, C.R.P. and De Faria, J.M. (1982) Application of information theory to sequential fault diagnosis. *IEEE Transactions on Computers*, **31**(2), 164–170.

Vitter, J.S. (1999) Online data structures in external memory. *Lecture Notes in Computer Science*, **1663**, 352–366.

Yeung, R.W. (1991) Alphabetic codes revisited. *IEEE Transactions on Information Theory*, **37**(3), 564–572.

## Appendix

The maximum number of internal nodes in a $(m+1)$-ary weight-balanced tree with $n$ terminal nodes (leaves) is denoted by $S_{\max}$ and is equal to:

$$S_{\max} = \left\lfloor \frac{n}{2m} \right\rfloor (m+1) + \delta,$$

where $\delta$

$$= \begin{cases} -1 & \text{if } n \bmod(2m) = 0, \\ 0 & \text{if } n \bmod(2m) = 1, \\ 1 & \text{if } 1 < n \bmod(2m) \leq m+1, \\ n \bmod(2m) - m & \text{if } m+1 < n \bmod(2m) \leq 2m-1. \end{cases}$$

## Biography

Irad Ben-Gal is an Assistant Professor in the Department of Industrial Engineering at Tel Aviv University. He holds a B.Sc. (1992) degree from Tel Aviv University, M.Sc. (1996) and Ph.D. (1998) degrees from Boston University. He is a member of the Institute for Operations Research and Management Sciences (INFORMS) and the Institute of Industrial Engineers (IIE). He is the head of the Computer Integrated Manufacturing (CIM) lab at Tel Aviv University. For several years he has worked for various industrial organizations as a consultant and a project manager. His research interests include, quality control, design-of-experiments, testing procedures, and application of information theory to industrial and bioinformatics problems.

*Contributed by the Reliability Engineering Department*