Optimal Detection of a Counterfeit Coin with Multi–Arms Balances*

Annalisa De Bonis
Department of Computer Science
Princeton University
Princeton, NJ 08544

Luisa Gargano
Dipartimento di Informatica ed Applicazioni
Università di Salerno
84081 Baronissi (SA), Italy

Ugo Vaccaro
Dipartimento di Informatica ed Applicazioni
Università di Salerno
84081 Baronissi (SA), Italy

Abstract

We consider the problem of locating a light coin out of a set containing n coins, $n-1$ of which have the same weight. The weighing device is a balance with $r \geq 2$ pans that, when r equally sized subset of coins are weighted, indicates the subset eventually containing the light coin. We give an algorithm to find the counterfeit coin that requires the minimum possible average number of weighings. All previous results on this problem considered two-arms balances only.

1 Introduction

The problem of locating a counterfeit coin out of a set of n coins, $n-1$ of which are good, is one of the oldest search problems ever studied and it is often used in introductory texts on the design and analysis of algorithms as a "paradigmatic example" (cfr. [15]). Many papers have been written on the counterfeit coin problem and several weighing models have been considered. For an account of the vast literature on the subject we refer the interested reader to [1], [2], [4], [10] and references therein quoted. Problems of searching for more than one coin have been studied in [3], [5], [6], [8], [11], [14], [16].

In one of the most popular models one is given a two–arms balance scale with which to compare the weights of two equally sized subsets of coins. The balance will tell us whether the subsets have the same weight, or the first set is lighter (and therefore contains the light coin), or

*The work was supported in part by the Italian Ministry of the University and of the Scientific Research in the framework of the "Algoritmi, Modelli di Calcolo e Strutture Informative" project.

the second one is lighter. The problem is to locate the counterfeit coin using as few weighings as possible. Two measures are commonly utilized to estimate the goodness of an algorithm: The worst–case number of weighings and the average number of weighings needed to locate the counterfeit coin; in the latter case it is assumed that one is given a probability distribution $\mathbf{p} = (p_1, \ldots, p_n)$, where p_i is the probability that the i–th coin is counterfeit[1]. Moreover, two classes of algorithms are usually considered: *Sequential* (or *adaptive*) algorithms and *predetermined* (or *non–adaptive*) algorithms. In sequential algorithms the weighing performed on the i–th step depends upon the feedbacks (outcomes) of the previous $i-1$ weighings, while in predetermined algorithms all the weighings are fixed beforehand (for more on these questions see [1]).

Sequential and predetermined optimal algorithms that locate a counterfeit coin using the minimum *worst–case* number of weighings are presented in ([1], Ch. 2). Optimal sequential algorithms requiring the *minimum average* number of weighings are given in [13] and [17], whereas Linial and Tarsi [12] find average-case optimal algorithms, both for the sequential and predetermined cases, for a variant of the classical model. More precisely, in [12] it is assumed that the counterfeit coin can be either lighter or heavier and this is not known *a priori.*

All above papers considered two–arms balances only. As an interesting generalization, Aigner [1] has proposed the problem of considering an r–arms balance scale ($r \geq 2$) such that when r equally sized subsets of coins are weighted in parallel, it indicates which subsets, if any, contains the lighter coin. Karp *et al.* [9] investigated the closely related problem of locating a given number k of defectives, but they considered a much powerful device that is able to weigh in parallel *any* r subsets of coins (not necessarily equally sized); the outcome indicates which of the subsets contain at least one defective coin. In his book Aigner [1] presents an optimal sequential algorithm requiring minimum worst–case number of weighings for the r–arms balance scale and states as an open problem that of finding an optimal sequential algorithm requiring minimum average number of weighings. In this paper we solve this problem thus also generalizing the result of [17] from $r = 2$ to arbitrary r.

We also show that the uniform distribution is the "worst" possible probability distribution on the set of coin, in the sense that for any other probability distribution the minimum average number of weighings is upper bounded by the minimum average number of weighings necessary when the uniform probability distribution is assumed.

[1]In this paper we assume, as done also in [12], [13] and [17], that the probability distribution $\mathbf{p}$ is the uniform one; however see also Section 3.

2 An Optimal Algorithm

In this section we present a sequential algorithm that locates a counterfeit coin out of n coins and requires the minimum possible average number of weighings.

We first establish the basic notation. We call a tree $(r+1)$–ary if each node has at most $r+1$ sons, called 0–son, 1–son,. . ., r–son, respectively. In the sequel all trees will be $(r+1)$–ary trees. Given a tree T we indicate by $T^{(i)}$ the subtree of T rooted at the i–son of the root of T, $i = 0, \ldots, r$, and by $|T|$ the number of leaves of T. Let us denote the set of coins by the set of natural numbers $S = \{1, \ldots, n\}$.

An algorithm to solve the counterfeit coin problem can be represented by a tree T whose root corresponds to the initial search space S and whose leaves correspond to the n coins; each internal node of T corresponds to a subset of S in a way that we explain below. We shall denote by

$$A_1 : \cdots : A_r, \qquad A_i \subset S, \text{ for } i = 1, \ldots, r$$

the comparison of the weights of the subsets of coins $A_1, \ldots, A_r$. Let us assume that after each weighing $A_1 : \cdots : A_r$ we receive a feedback i, with $i = 0$ if all weighted subsets of coins have equal weight, and $1 \leq i \leq r$ if the i-the subset A_i is lighter than the others, that is, if A_i contains the counterfeit coin. We denote by $S(i_1 \cdots i_k)$ the search space when the feedbacks of the first k weighings are $i_1 \cdots i_k$, that is, $S(i_1 \cdots i_k)$ is made by all coins $\{c_1, \ldots c_p\} \subseteq S$ for which the test feedbacks $i_1 \cdots i_k$ are consistent with the assumption that the counterfeit coin belongs to $\{c_1, \ldots c_p\}$. If $S(i_1 \cdots i_k) \neq \emptyset$ then the tree T contains a node labeled by $S(i_1 \cdots i_k)$ whose i–son exists and is labeled by $S(i_1 \cdots i_k i)$ if $S(i_1 \cdots i_k i) \neq \emptyset$, it does not exist otherwise. The possible outputs of the algorithm are the $n = |S|$ leaves of the tree, i.e., the nodes labeled by sets $S(i_1 \cdots i_k)$ with $|S(i_1 \cdots i_k)| = 1$.

Example 1 *Let the set of coins be $S = \{1, \ldots, 21\}$ and suppose we are given a 3-arms balance scale. An optimal algorithm that finds the counterfeit coin in S and uses the minimum average number of weighings is given in Fig. 1. The internal nodes of the tree represent the weighings performed in that step. If at a given node we weigh the subsets $A_1 : A_2 : A_3$, we assume that the i-th branch ($i = 0, 1, 2, 3$ counting from the left) corresponds to the event that the counterfeit coin belongs to the i-th set if $i \geq 1$, that it does not belong to any if $i = 0$. Boldface numbers represent coins that, due to the results of the previous weighings, are known to be not counterfeit and are used again in order to balance the pans of the scale. The necessity of using such coins will be made clear in Theorem 2. The search space at each node of the tree is the union of all subset of weighed coins, apart from the ones written in bold that are already known to be not counterfeit.*

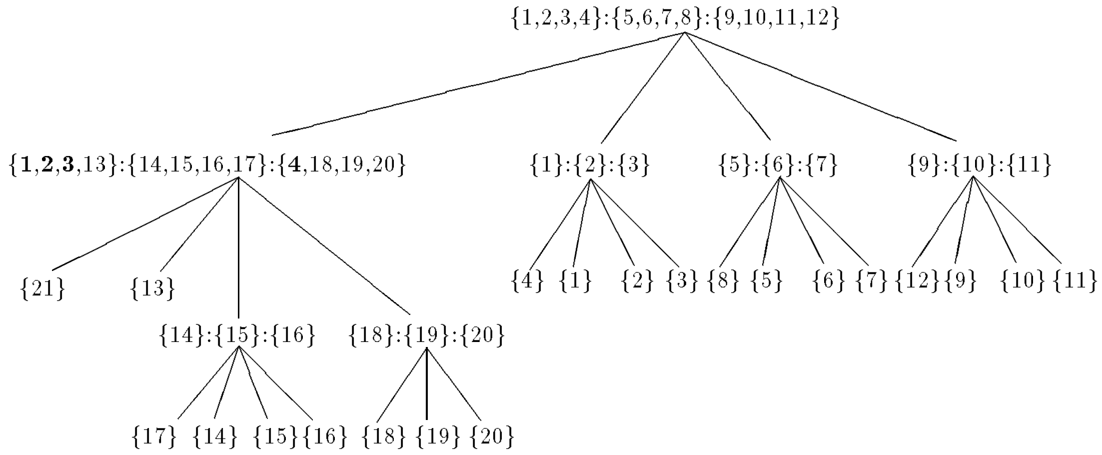


FIGURE 1

Given a tree T, let $h(x,T)$ represents the level of the leaf x in T, that is, the distance of x from the root of T. The external path length $h(T)$ of T is defined as $h(T) = \sum h(x,T)$ where the summation is taken over the n leaves of T. Under the hypothesis of uniform probability distribution on the n coins, the average number of weighings performed by an algorithm represented by a tree T is given by $h(T)/n$.

The problem of determining the quantity $H(n) = \min h(T)$, where the minimum is taken over *all* $(r+1)$–ary trees with n leaves is a special case of the well-known Huffman problem [7]. Essentially, in this paper we face the problem of finding the quantity $\min h(T)$ over a *restricted* class of $(r+1)$–ary trees, where the restrictions are determined by the testing device we are considering (cf., next Property 1).

Given an integer n with $(r+1)^L \le n < (r+1)^{L+1}$ we shall represent n as

$$n = (r+1)^L + kr + j, \qquad \text{for some } 0 \le k < (r+1)^L, \quad 0 \le j \le r-1.$$

The following result is classic (e.g., see [1]) and allows to find $H(n)$ explicitly.

Theorem 1 *Given an integer n, $n = (r+1)^L + kr + j$, where $0 \le k < (r+1)^L$, $0 \le j \le r-1$, a tree T has external path length $h(T)$ equal to $H(n)$ if and only if T has $n - \lceil (kr+j)(r+1)/r \rceil$ leaves at level L and $\lceil (kr+j)(r+1)/r \rceil$ at level $L+1$. Moreover,*

$$H(n) = nL + \left\lceil (kr+j)\frac{r+1}{r} \right\rceil = n\lfloor \log_{r+1} n \rfloor + \left\lceil \left(n - (r+1)^{\lfloor \log_{r+1} n \rfloor}\right) \frac{r+1}{r} \right\rceil$$

Let T_L be the tree with $(r+1)^L$ leaves at level L. A tree with n leaves and having external path length equal to $H(n)$ can be obtained from T_L by changing k leaves into internal nodes each having $r+1$ sons if $j = 0$ and, if $j > 0$, one more leaf into an internal node having $j+1$ sons.

While any algorithm to solve the counterfeit coin problem can be represented by a tree, the contrary is not true. In fact we have the following result.

Property 1 *A tree T represents a search algorithm only if for each $i = 1, \ldots, r$*

$$|T^{(1)}| = |T^{(2)}| = \cdots = |T^{(r)}| > 0 \tag{1}$$

where $|T^{(i)}|$, $i = 1, \ldots, r$, denotes the number of leaves in the subtree $T^{(i)}$.

Proof The necessity of (1) is immediate once we notice that $|T^{(i)}|$ corresponds to the size of the set of coins we weight on the i–th pan at the first step of the algorithm. □

We call a tree T *admissible* if there exists an algorithm $\mathcal{A}$ that solves the counterfeit coin problem such that T represents $\mathcal{A}$. If the set of coins has cardinality $n = (r+1)^L$ then, trivially, the tree T_L is both admissible and optimal; the corresponding algorithm weighs at i-th step subsets of coins of size $(r+1)^{L-i}$, for any $1 \leq i \leq L$. We state explicitly this result for future reference.

Lemma 1 *If $n = (r+1)^L$ then optimal algorithms correspond to the tree T_L with $h(T_L) = H(n)$.*

The model under study imposes that at each step the algorithm must put the same number of coins on each pan of the scale. However, if the first test gives feedback i, at least $n - |T^{(i)}|$ coins are known to be standard (i.e., not counterfeit) and can eventually be used to balance the number of coins in each pan during the successive weighings, (cfr. [1], [17]). This observation, together with Lemma 1, allows us to derive the desired result on the minimum possible average number of weighings, denoted by $\overline{L}(n)$, required by any sequential search algorithm on n coins. Recalling the obvious lower bound $\overline{L}(n) \geq H(n)/n$ we have the following theorem that represents our main result.

Theorem 2 *For each $n = (r+1)^L + kr + j \geq r$, where $0 \leq k < (r+1)^L$ and $0 \leq j \leq r-1$ one has*

$$\overline{L}(n) = \frac{H(n)}{n} + \begin{cases} \frac{j}{n} & \text{if } 2(r+1) \leq n \leq 3r, \\ \frac{r-k}{n} & \text{if } 3r+1 \leq n \leq r(r+1), \\ 0 & \text{otherwise.} \end{cases}$$

Proof Given $n = (r+1)^L + kr + j \geq r$, we will prove the existence of an admissible tree T with n leaves having external path length

$$H(n) + \begin{cases} j & \text{if } 2(r+1) \leq n \leq 3r, \\ r-k & \text{if } 3r+1 \leq n \leq r(r+1), \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and show that if $2(r+1) \leq n \leq r(r+1)$ then *any* admissible tree has external path length not less than that in (2).

If $L = 0$, i.e., $n = r$, (2) is trivially true. We consider separately the case $L = 1$ and the case $L \geq 2$.

Consider first $L = 1$.

1) If $n = 2r+1$ or $n = r+1+j$, with $0 \leq j \leq r-1$, admissible trees having external path length equal to $H(n)$ are shown in Fig.2(a) and Fig.2(b), respectively.

2) Let us consider now $n = r+1+kr+j$ with $2r+2 \leq n \leq r(r+1)$. ¿From Property 1 the optimal admissible tree must be searched among the trees on n nodes such that

$$|T^{(1)}| = |T^{(2)}| = \ldots = |T^{(r)}| = i, \tag{3}$$

for some $1 \leq i \leq k+1$.

If $i = k+1$ the only possible admissible tree is shown in Fig.3(a) if $j = 0$ and in Fig.3(b) if $j \geq 1$. Such a tree has external path length $H(n) + (r-k)$.

If $2 \leq i \leq k$ it is immediate to see that in any tree satisfing (3) each leaf must have level not less than 2. Therefore the external path length of the tree cannot be less than $H(n) + (r-k)$.

If $i = 1$ we can write

$$\begin{aligned} h(T) &= r + \left[(n-r) + h\left(T^{(0)}\right)\right] = n + H(n-r) + \left[h\left(T^{(0)}\right) - H(n-r)\right] \\ &= n + H(n) - 2r - 1 + \left[h\left(T^{(0)}\right) - H(n-r)\right] \end{aligned}$$

In case $k = 1$ we can take $T^{(0)}$ as the optimal tree on $n - r = r+1+j$ leaves (see case 1)) and have $h(T) = H(n)+j$. On the other hand, if $k > 1$ one has $h(T) \geq H(n)+n-2r-1 = H(n) + r(k-1) + j > H(n) + r - k$.

We can then conclude that if $k = 1$ the optimal tree is obtained for $i = 1$ and has external path length $H(n) + j$, if $k > 1$ the optimal tree is obtained for $i = k+1$ and has external path length $H(n) + r - k$.

Consider now $n = (r+1)^L + kr + j$ with $L \geq 2$ and define

$$\ell(n) = \begin{cases} k & \text{if } j = 0, \\ k+1 & \text{if } j > 0. \end{cases}$$

Consider the tree T_L with $(r+1)^L$ leaves at level L; we will obtain from T_L an admissible tree T with n leaves. This is done by changing $\ell(n)$ leaves of T_L into internal nodes, k of which having $r+1$ sons and, if $\ell(n) = k+1$, the additional one having $j+1$ sons. Notice that, by Theorem 1, the resulting tree T has external path length $h(T)$ equal to $H(n)$, i.e., the minimum possible. Recall that $T^{(i)}$, $i = 0, 1, \ldots, r$, is the subtree of T rooted in the i–son of the root of T.
Depending on the value of $\ell(n)$, we distinguish three cases for the choice of the leaves to be changed into internal nodes.

i) If $\ell(n) \geq r+1$ and $\ell(n) > r\lceil \ell(n)/(r+1)\rceil$ we choose $\lceil \ell(n)/(r+1)\rceil$ leaves in each subtree $T_L^{(i)}$, for $i = 1, \ldots, r$, and the remaining $\ell(n) - r\lceil \ell(n)/(r+1)\rceil$ in $T_L^{(0)}$; to maintain Property 1, the eventual leaf to which we append $j+1$ sons is a leaf in $T_L^{(0)}$.

ii) If $\ell(n) \geq r+1$ and $\ell(n) \leq r\lceil \ell(n)/(r+1)\rceil$ we choose $\lfloor \ell(n)/(r+1)\rfloor$ leaves in each $T_L^{(i)}$, for $i = 1, \ldots, r$, and $\ell(n) - r\lfloor \ell(n)/(r+1)\rfloor$ in $T_L^{(0)}$; to maintain Property 1, the eventual leaf to which we append $j+1$ sons is a leaf in $T_L^{(0)}$.

iii) If $\ell(n) \leq r$ we choose all the $\ell(n)$ leaves in the subtree $T_L^{(0)}$.

Call T the resulting tree. The rest of the proof is devoted to show that T is admissible. Let $\mathcal{A}$ be an algorithm that searches a space S of size $|S| = (r+1)^L$ and is represented by T_L. We will describe an algorithm $\mathcal{A}'$ that searches a space S' of size $|S'| = |T| = n$ and is represented by T.
Indicate by $x_1, \ldots, x_{\ell(n)}$ the coins that label the leaves of T_L and have been transformed into internal nodes to obtain T. Moreover, let f_i be the number of sons of x_i in T, for $i = 1, \ldots, \ell(n)$. We can see the transformation of T_L into T as the substitution of each coin (leaf) x_i with a "super"–coin $X_i = \{x_{i,1}, \ldots, x_{i,f_i}\}$. The algorithm $\mathcal{A}'$ is described as follows: at each step, if $\mathcal{A}$ weights x_i, then $\mathcal{A}'$ substitutes x_i with the "super"–coin X_i, for $i = 1, \ldots, \ell(n)$.

Given a set of coins $A \subseteq S$ the substitution of coins $x_1, \ldots, x_{\ell(n)}$ with the "super"–coins $X_1, \ldots, X_{\ell(n)}$ transforms of A into the set

$$B(A) = A - \{x_t \ : x_t \in A,\ 1 \leq t \leq \ell(n)\} \cup \left(\bigcup_{\substack{1 \leq t \leq \ell(n) \\ x_t \in A}} X_t \right). \tag{4}$$

Let us consider the initial step: Algorithm $\mathcal{A}$ performs a weighing $A_1 : A_2 : \cdots : A_r$ with $|A_i| = |T_L^{(i)}| = (r+1)^{L-1}$, for $i = 1, \ldots, r$. The new algorithm $\mathcal{A}'$ performs the weighing

$B(A_1) : B(A_2) : \cdots : B(A_r)$ with $|B(A_i)| = |T^{(i)}|$, for $i = 1, \ldots, r$. We recall that this is an admissible weighing since rules i) , ii) and iii) respect Property 1 and therefore imply $|T^{(1)}| = \ldots = |T^{(r)}|$.

To consider successive weighings, notice that if the first step gives feedback i for some $0 \leq i \leq r$, then

$$\sum_{\substack{0 \leq t \leq r \\ t \neq i}} |T^{(t)}| \geq \sum_{\substack{0 \leq t \leq r \\ t \neq i}} |T_L^{(t)}| = (r+1)^{L-1} r \tag{5}$$

coins are known to be standard and can be used to balance the pans in successive weighings.

Consider a node in T_L labeled with the search space $S_{\mathcal{A}}(i_1 \cdots i_h)$, $h \geq 1$, corresponding in the algorithm $\mathcal{A}$ to the sequence of feedbacks $i_1 \cdots i_h$. If $S_{\mathcal{A}}(i_1 \cdots i_h)$ is an internal node in T_L let $C_1 : C_2 : \cdots : C_r$ be the weighing step done by $\mathcal{A}$ at that node. For each $i = 1, \ldots r$ consider the set $B(C_i)$ defined by (4). Notice that sets $B(C_i)$'s might have different sizes and therefore we cannot simply substitute the C_i's with the $B(C_i)$'s. In order to get subsets of coins of equal size that can be weighted with our balance scale, we use coins that are known to be standard at this step of the algorithm. Therefore, we define new sets C_i' as obtained from $B(C_i)$ by adding $\max_{1 \leq t \leq r} |B(C_t)| - |B(C_i)|$ standard coins. After that addition of known standard coins the new algorithm $\mathcal{A}'$ will perform the weighing $C_1' : C_2' : \cdots : C_r'$. Therefore, the total number of standard coins required by the weighing $C_1' : C_2' : \cdots : C_r'$ is at most

$$\left(\max_{1 \leq t \leq r} |B(C_t)| - \min_{1 \leq t \leq r} |B(C_t)| \right) (r-1) \leq (r+1)^{L-h-1} r(r-1) \leq (r+1)^{L-2} r(r-1).$$

Finally, if $S_{\mathcal{A}}(i_1 \cdots i_h)$ is a leaf of T_L with $S_{\mathcal{A}}(i_1 \cdots i_h) = \{x_i\}$, for some $1 \leq i \leq \ell(n)$, the search space of $\mathcal{A}'$ corresponding to the sequence of feedbacks $i_1 \cdots i_h$ is the "super"–coin X_i, that is $S_{\mathcal{A}'}(i_1 \cdots i_h) = \{x_{i,1}, \ldots, x_{i,f_i}\}$. In this case $\mathcal{A}'$ performs the last weighing $x_{i,1} : \cdots : x_{i,f_i}$ if $f_i = r$, the last weighing $x_{i,1} : \cdots : x_{i,f_i - 1}$ if $f_i = r+1$, the last weighing $x_{i,1} : \cdots : x_{i,f_i} : s_1 : \cdots : s_{r-f_i}$, where $s_1, \ldots, s_{r-f_i}$ are standard coins, otherwise.

In each case the number of standard coins necessary to equalize the contents of the pans is less than the number of available standard coins (see (5)) and the theorem is proved. □

Remark Above Theorem 2 includes, as particular case when $r = 2$, the main result of [17]. Moreover, Theorem 2 shows that the optimal algorithm with respect to the average-case number of weighings is *not* optimal with respect to the worst-case number of weighings. Indeed, Aigner [1] shows that the optimal worst-case algorithm is always represented by a tree that has all leaves on at most two levels. Theorem 2 shows that for some particular values of the number of leaves (coins), the optimal average-case algorithm corresponds to a tree that has leaves on three different levels. It is worth pointing out that this phenomenon occurs only when the number of pans r is greater than or equal to three.

3 Arbitrary Probability Distributions

In the previous section we have established the values of $\overline{L}(n)$, the minimum average number of steps of any sequential algorithm to search among n coins, under the assumption of uniform probability distribution on the set of coins. We show now that if an *arbitrary* probability distribution on the set of coins is assumed then the minimum average number of weighings required to search among n coins is upper bounded by $\overline{L}(n)$

Let $S = \{c_1, \ldots, c_n\}$ be the set of n coins and $\mathbf{p} = (p_1, \ldots, p_n)$ be a probability distribution on S. Let $\overline{L}(\mathbf{p}, n)$ denote the minimum average number of weighings required by any sequential algorithm to search S under the assumption that the coin c_i is counterfeit with probability p_i, $i = 1, \ldots, n$. With this new notation we have that $L(n) = L(\mathbf{U}_n, n)$, where $\mathbf{U}_n = (1/n, \ldots, 1/n)$ is the uniform probability distribution with n components. The following result holds.

Theorem 3 *For each number of coins n and probability distribution* $\mathbf{p}$

$$\overline{L}(\mathbf{p}, n) \leq \overline{L}(n). \tag{6}$$

Proof Let T be a tree with n leaves labeled $c_1, \ldots, c_n$. We recall that $h(c_i, T)$ is the level of the leaf c_i in T. Suppose that T represents some algorithm $\mathcal{A}$ that locates the counterfeit coin among $\{c_1, \ldots, c_n\}$ using an r-arms balance, i.e., T satisfies Property 1. The average number of weighings made by the algorithm $\mathcal{A}$ represented by the tree T is $h(T, \mathbf{p}) = \sum_{i=1}^{n} h(c_i, T)p_i$. The quantity $\overline{L}(\mathbf{p}, n)$ can be written as

$$\overline{L}(\mathbf{p}, n) = \min h(T, \mathbf{p}),$$

where the minimum is taken among all admissible trees (i.e. representing some algorithm).

Consider now an optimal admissible tree Λ for the uniform distribution $\mathbf{U}_n$, that is, Λ is such that $\overline{L}(n) = h(\Lambda, \mathbf{U}_n)$. We will show

$$h(\Lambda, \mathbf{p}) \leq h(\Lambda, \mathbf{U}_n). \tag{7}$$

This proves the lemma since it implies $\overline{L}(\mathbf{p}, n) = \min h(T, \mathbf{p}) \leq h(\Lambda, \mathbf{p}) \leq h(\Lambda, \mathbf{U}_n) = \overline{L}(n)$.

Let us then prove inequality (7). Consider first the case $n \notin \{2(r+1), \ldots, 3r\}$. By Theorems 1 and 2 we know that the level of each leaf of Λ is either $\lfloor \log_{r+1} n \rfloor$ or $\lceil \log_{r+1} n \rceil$. Notice now that if two probabilities p_i and p_j of $\mathbf{p}$ satisfy the relation $p_i \geq p_j$ then the assignement of coins to the leaves of Λ which minimizes $h(\Lambda, \mathbf{p}) = \sum_{i=1}^{n} h(c_i, \Lambda) p_i$ satisfies $h(c_i, \Lambda) \leq h(c_j, \Lambda)$. Since we are interested in upper bounding $h(\Lambda, \mathbf{p})$ we can assume without loss of generality that $c_1, \ldots, c_\ell$ are the leaves of Λ at level $\lfloor \log_{r+1} n \rfloor$ and $p_i \geq p_j$ for each $i = 1, \ldots, \ell$ and $j = \ell + 1, \ldots, n$
We have

$$\begin{aligned} h(\Lambda, \mathbf{p}) &= \sum_{i=1}^{n} h(c_i, \Lambda) p_i = \sum_{i=1}^{\ell} \lfloor \log_{r+1} n \rfloor p_i + \sum_{i=\ell+1}^{n} \lceil \log_{r+1} n \rceil p_i \\ &= \lceil \log_{r+1} n \rceil - (\lceil \log_{r+1} n \rceil - \lfloor \log_{r+1} n \rfloor) \sum_{i=1}^{\ell} p_i \\ &\leq \lceil \log_{r+1} n \rceil - (\lceil \log_{r+1} n \rceil - \lfloor \log_{r+1} n \rfloor) \frac{\ell}{n}, \end{aligned}$$

where the last inequality holds since $p_i \geq p_j$ for each $i = 1, \ldots, \ell$ and $j = \ell + 1, \ldots, n$. Therefore

$$\begin{aligned} h(\Lambda, \mathbf{U}_n) &= \sum_{i=1}^{n} h(c_i, \Lambda)/n = \sum_{i=1}^{\ell} \lfloor \log_{r+1} n \rfloor / n + \sum_{i=\ell+1}^{n} \lceil \log_{r+1} n \rceil / n \\ &= \lceil \log_{r+1} n \rceil - (\lceil \log_{r+1} n \rceil - \lfloor \log_{r+1} n \rfloor) \frac{\ell}{n} \\ &\geq h(\Lambda, \mathbf{p}) \end{aligned}$$

Finally, consider the case $2(r+1) \leq n \leq 3r$. From Theorem 2 we know that Λ has r leaves (say $c_1, \ldots, c_r$) at level 1, r leaves (say $c_{r+1}, \ldots, c_{2r}$) at level 2, and $n - 2r$ (say $c_{2r+1}, \ldots, c_n$) at level 3. Again we can assume, without loss of generality, that $p_i \geq p_s \geq p_t$ for $i = 1, \ldots, r$, $s = r+1, \ldots, 2r$, and $t = 2r+1, \ldots, n$. We have

$$\begin{aligned} h(\Lambda, \mathbf{p}) &= \sum_{i=1}^{n} h(c_i, \Lambda) p_i = \sum_{i=1}^{r} p_i + \sum_{i=r+1}^{2r} 2p_i + \sum_{i=2r+1}^{n} 3p_i \\ &= 3 - \sum_{i=1}^{r} p_i - \sum_{i=1}^{2r} p_i \\ &\leq 3 - \frac{r}{n} - \frac{2r}{n} \\ &= 3 - \frac{3r}{n} \end{aligned}$$

and

$$h(\Lambda, \mathbf{U}_n) = \frac{r}{n} + 2\frac{r}{n} + 3\frac{n-2r}{n} = 3 - \frac{3r}{n} \leq h(\Lambda, \mathbf{p})$$

that proves (7). Hence the theorem. $\square$

References

[1] M. Aigner, *Combinatorial Search*, Wiley–Teubner, New York–Stuttgart (1988).

[2] R. Bellman and B. Glass, "On Various Versions of the Defective Coin Problem", *Inform. & Control* 4, (1961), 118–151.

[3] X. M. Chang, F. K. Hwang and J. F. Weng, "Optimal Detection of Two Defectives with a Parity Check Device", *SIAM J. Disc. Math.* 1, (1988), 38–44.

[4] C. Christen and F. K. Hwang, "Detection of a Defective Coin with Partial Weight Information", *Amer. Math. Monthly* 91, (1984), 173–179.

[5] L. Gargano, J. Körner and U. Vaccaro, "Search Problems for Two Irregular Coins with Incomplete Feedback: the Underweight Model", *Discrete Appl. Math.* 36, (1992), 191–197.

[6] L. Gargano, V. Montuori, G. Setaro, and U. Vaccaro, "An Improved Algorithm for Quantitative Group Testing", *Discrete Appl. Math.* 36, (1992), 299–306.

[7] D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes", *Proc. IRE* 40, (1952), 1098–1101.

[8] F. K. Hwang, "A Tale of Two Coins", *Amer. Math. Monthly* 94, (1987), 121–129.

[9] R. M. Karp, E. Upfal, A. Wigderson, "The Complexity of Parallel Search", *J. Comput. and System Sci.* 36, (1988), 225–253.

[10] G. O. H. Katona, "Combinatorial Search Problems", in: *A Survey of Combinatorial Theory*, J. N. Srivastava et al., Eds., North–Holland, (1973), 285–308.

[11] Ker–I Ko, "Searching for Two Objects by Underweight Feedback", *SIAM J. Disc. Math.* 1, (1988), 65–70.

[12] N. Linial and M. Tarsi, "The Counterfeit Coin Problem Revisited", *SIAM J. Comp.* 11, (1982), 409–415.

[13] D. G. Mead, "The Average Number of Weighings to Locate a Counterfeit Coin", *IEEE Trans. Inform. Theory* 25, (1979), 616–617.

[14] L. Pyber, "How to Find Many Counterfeit Coins", *Graph and Combinatorics* 2, (1986), 173–177.

[15] E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice*, Prentice–Hall, Englewood Cliffs, NJ, (1977).

[16] R. Tŏsić, "Two Counterfeit Coins", *Discrete Math.* 46, (1983), 295–289.

[17] K. Winkelmann, "An Improved Strategy for a Counterfeit Coin Problem", *IEEE Trans. Inform. Theory* 28, (1982), 120–122.

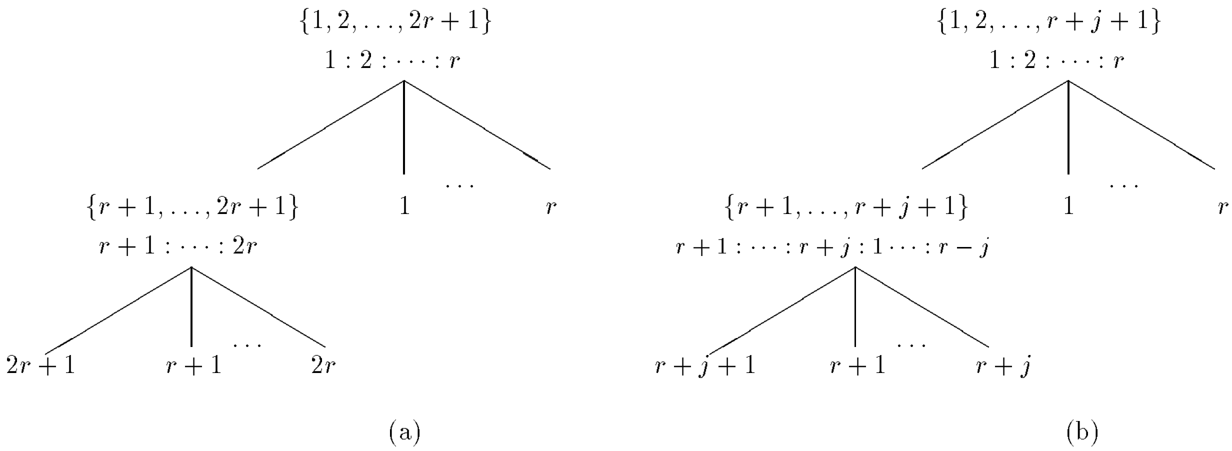
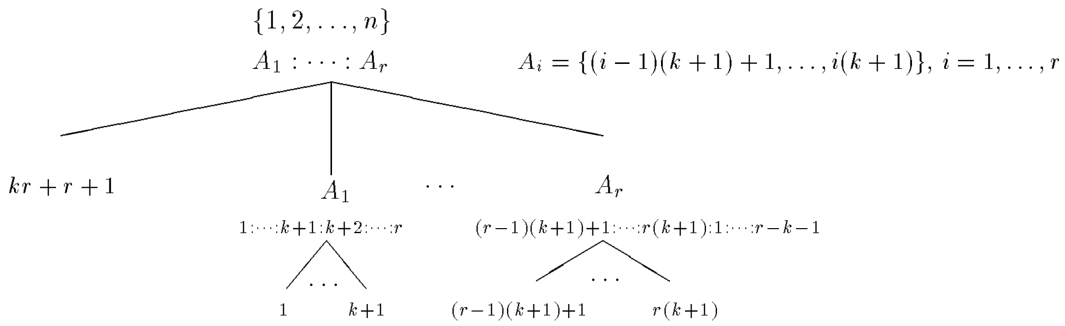
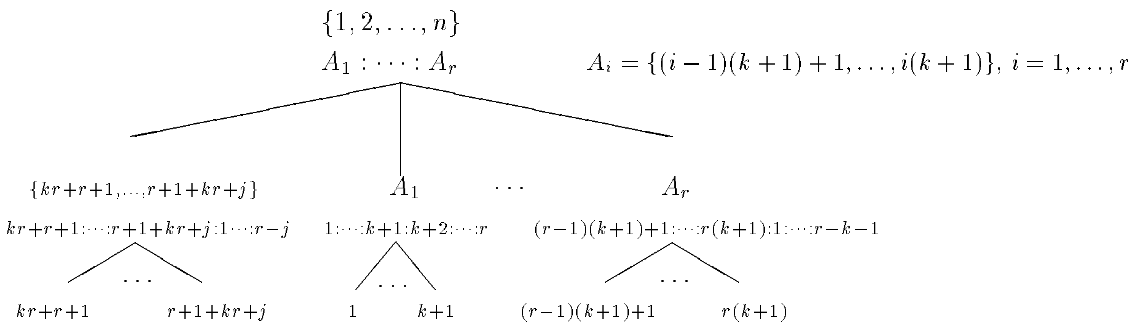

$\{1, 2, \ldots, 2r+1\}$
$1 : 2 : \cdots : r$
$\{r+1, \ldots, 2r+1\}$
$r+1 : \cdots : 2r$
1
$\cdots$
r
$2r+1$
$r+1$
$\cdots$
$2r$
(a)
$\{1, 2, \ldots, r+j+1\}$
$1 : 2 : \cdots : r$
$\{r+1, \ldots, r+j+1\}$
$r+1 : \cdots : r+j : 1 \cdots : r-j$
1
$\cdots$
r
$r+j+1$
$r+1$
$\cdots$
$r+j$
(b)

FIGURE 2


$\{1,2,\ldots,n\}$
$A_1:\cdots:A_r$
$A_i=\{(i-1)(k+1)+1,\ldots,i(k+1)\},\ i=1,\ldots,r$
$kr+r+1$
A_1
$\cdots$
A_r
$1:\cdots:k+1:k+2:\cdots:r$
$(r-1)(k+1)+1:\cdots:r(k+1):1:\cdots:r-k-1$
1
$k+1$
$(r-1)(k+1)+1$
$r(k+1)$

(a)


$\{1,2,\ldots,n\}$
$A_1:\cdots:A_r$
$A_i=\{(i-1)(k+1)+1,\ldots,i(k+1)\},\ i=1,\ldots,r$
$\{kr+r+1,\ldots,r+1+kr+j\}$
A_1
$\cdots$
A_r
$kr+r+1:\cdots:r+1+kr+j:1\cdots:r-j$
$1:\cdots:k+1:k+2:\cdots:r$
$(r-1)(k+1)+1:\cdots:r(k+1):1:\cdots:r-k-1$
$kr+r+1$
$r+1+kr+j$
1
$k+1$
$(r-1)(k+1)+1$
$r(k+1)$

(b)

FIGURE 3