Optimal Algorithms for the Coin Weighing Problem with a Spring Scale

Nader H. Bshouty*
Technion, Israel
bshouty@cs.technion.ac.il

Abstract

Suppose we are given n coins out of a collection of coins of two distinct weights w_0 and w_1, true and counterfeit coins, respectively, where d of them are counterfeit coins. Assume we are allowed to weigh subsets of coins in a spring scale. Determine the counterfeit coins in a minimal number of weighing.

This problem is equivalent to the following learning problem: Given a linear function $f = x_{i_1} + x_{i_2} + \cdots + x_{i_d}$, where $1 \le i_1 < i_2 < \cdots < i_d \le n$ and a substitution oracle of values in the domain $\{0,1\}^n$ to f. Find f with minimal number of substitution queries.

In this paper we give the first optimal[1]. (in the number of weighing or substitutions) *polynomial time* adaptive algorithm that determines the counterfeit coins.

We then extend our algorithm to the following more general coin weighing problems with a spring scale: Suppose we are given n coins out of a collection of coins of unknown integer weights. Determine the weight of each coin in a minimal number of weighing. We give an optimal adaptive polynomial time algorithm for this problem. This algorithm is based on a new optimal adaptive algorithm for reconstructing bounded weight vectors in polynomial time. This solves the general problem of learning *any* linear function with bounded integer coefficient in polynomial time with optimal number of substitution queries.

To the best of our knowledge all the algorithms in this paper are the first optimal polynomial time adaptive algorithms for the problem of coin weighing in a spring scale.

Keywords: Computational Learning, Combinatorial Search Problem.

*This research was done while the author was visiting Google in Mountain View, California

[1]In this paper optimal will mean $\le 2\times$ (lower bound). In very few cases it means $\le 5\times$ (lower bound)

1 Introduction

The *coin weighing problem with a spring scale* is the following [S60]: Suppose we are given n coins out of a collection of coins of two distinct weights w_0 and w_1, true and counterfeit coins, respectively.[2] Assume we are allowed to weigh subsets of coins in a spring scale. Determine the weight of each coin in a minimal number of weighing.

This problem has many applications and is also known as the detection problem [SS63], determining collection [CM66], counterfeit coins problem, distinguishing family problem [LV91] and rigidity of the n-dimensional Hamming space problem [KLT00] and is equivalent to the uniquely decodable codes for noiseless n-user adder channel [CW79, MK87] and the Mastermind game with two colors [K76]. See also [CK08] for other applications for this problem.

A non-adaptive algorithm for the coin weighing problem is equivalent to finding a $k \times n$ (0,1)-matrix, M such that for every $u, v \in \{0,1\}^n$ where $u \ne v$ we have $Mv \ne Mu$. The weighing complexity is k, the number of rows of M. Such a matrix is called a search matrix. For the non-adaptive algorithm to run in polynomial time we need that a search matrix be built in polynomial time and given Mv where $v \in \{0,1\}$, one can find v in polynomial time.

This problem was first introduced by Shapiro in 1960 for $n = 5$ [S60] and was solved (for $n = 5$) by Fine and many others [F60]. It was then studied for any n by Cantor [C62], Soderberg and Shapiro [SS63], Erdös and Rényi [ER63], Lindström [L65, L66, L71, L75] and Cantor and Mills [CM66]. See also [A88, MK87, DH93]. The information theoretic lower bound for the number of weighing is (asymptoticly)[3]

$$\frac{n}{\log n}$$

that applies even for adaptive algorithms. Erdös and Rényi [ER63] and independently, Moser [M70] (see [L65]) proved the lower bound

$$k \ge \frac{2n}{\log n}$$

[2]When w_1 and w_2 are not known, we can find them adaptively in $1 + \lceil \log n \rceil$ weighings [CH84]. When w_1 and w_2 are known the problem can be reduced to the case $w_0 = 0$ and $w_1 = 1$.

[3]Here and throughout the paper we will only write the asymptotic weighing complexity of the problem. So by asymptotic bound C we mean $C(1 \pm o(1))$.

for any $k \times n$ search matrix. See also [CM66, L66, M70, L75, P77, A88, LV91]. This shows that any non-adaptive algorithm for the coin weighing problem must make at least $2n/\log n$ weighing.

Lindström [L64, L65] and, independently, Cantor and Mills [CM66] presented a construction of a search matrix of size $k \times n$ where k is

$$\frac{2n}{\log n}.$$

Then using the theory of Möbius functions, Lindström [L71] gave another construction with the same asymptotic bound. See also a simple construction in [L75, A88] and a simple recursive construction in [CM66, MK87]. Although not explicitly indicated by the above papers, the search matrix M in Lindström construction and Cantor and Mills construction can be constructed in polynomial time. In Lindström constructions [L64, L65, L71] and in Cantor and Mills construction [CM66, MK87], v can be extracted from Mv in polynomial time.

We give a new simple construction that is based on Fourier transform. The advantage of this construction is that we can use Fourier transform to extract v from Mv. Another advantage is that our construction can be easily generalized to many other search matrices that can handle other coin weighing problems.

When one allows adaptiveness, one can find the number of counterfeit coins d (by finding w_0 and w_1 and then weighing all the coins) and then solves the following problem.

The *d-coin weighing problem with a spring scale* is the following [D75, L75, C80, AS85, A86, A88]: Suppose we are given n coins out of a collection of coins of two distinct weights w_0 and w_1, true and counterfeit coins, respectively, where d of them are counterfeit coins. Assume we are allowed to weigh subsets of coins in a spring scale. Determine the weight of each coin in a minimal number of weighing.

The information theoretic lower bound for this problem gives[4]

$$\frac{d \log \frac{n}{d}}{\log d}$$

weighing for any adaptive algorithm. For non-adaptive algorithms, Djackov [D75] and Lindström, [L75] showed that any non-adaptive algorithm must make at least

$$\frac{2d \log \frac{n}{d}}{\log d}$$

weighings. Then using the Kronecker product of matrices he gave an optimal non-adaptive algorithm that runs in polynomial time for the case when we have d sets of coins each of size n/d and each set contains exactly one counterfeit coin. Grebinski and Kucherov [GK00] showed non-constructively that a non-adaptive algorithm exists that asks

$$\frac{4d \log \frac{n}{d}}{\log d}$$

weighings[5]. They show that a randomized 0-1 matrix of such size is a search matrix for this problem with non-zero probability. Their algorithm does not lead to a randomized polynomial time algorithm since it is not clear how to find the counterfeit coins from the results of the weighings.

The best polynomial time non-adaptive algorithm known for the d-coin weighting problem makes

$$d \log n$$

weighings [L72, L75]. See also [H99] for any d and [A88] for $d = 2$ that can be extended to any d. The best polynomial time adaptive algorithm known makes

$$d \log \frac{n}{d}$$

weighings [C79, Ca79, TM78, M81, UTW00] and chapter 10 in [DH93] . Those algorithms are asymptotically optimal only when d is constant. Many papers in the literature studied the problem for $d = 2$ [L71, L72, L75, C80, C83, A86, A88, H90, CLZ01].

In this paper we give an adaptive polynomial time algorithm that makes

$$\frac{2d \log \frac{n}{d}}{\log d} + O\left(\frac{d}{\log d} + \frac{d(\log\log d) \log \frac{n}{d}}{(\log d)^2}\right).$$

weighings. To the best of our knowledge this is the first polynomial time optimal algorithm for this problem.

Our algorithm makes use of the following extension of search matrices. A $(d_1, d_2, \ldots, d_n)$-detecting matrix M is a $(0,1)$-matrix such that for every $v, u \in \prod_i \{0, 1, \ldots, d_i - 1\}$, $v \neq u$ we have $Mv \neq Mu$. Detecting matrices are known from the literature only when $d_i = d$ for all i [L71]. We extend our construction to build an optimal $(d_1, d_2, \ldots, d_n)$-detecting matrix in polynomial time. We then use this construction to show a new divide and conquer approach. We divide the problem into t disjoint sub-problems and using the above construction solve all the problems in $O(t/\log t)$ times the average weighing complexity of the sub-problems.

We then extend our algorithm to the following more general coin weighing problems with a spring scale: Suppose we are given n coins out of a collection of coins of unknown integer weights. Assume we are allowed to weigh subsets of coins in a spring scale. Determine the weight of each coin in a minimal number of weighing.

An information-theoretic argument gives the lower bound

$$k(n, W) = \begin{cases} \frac{W \log\left(\frac{n}{W}+1\right)}{\log W} & W \leq n \\ \frac{n \log\left(\frac{W}{n}+1\right)}{\log n} & n < W \leq n^2 \\ n & W > n^2 \end{cases} \quad (1)$$

for the number of weighing where W is the sum of the weights.

This problem was studied for $W \leq n$ in [P81, RV97, GK00] and for $W > n$ in [GK00]. Pippenger [P81] showed nonconstructively that for $W = n$ there is a non-adaptive algorithm that makes $O(W/\log W)$ weighings. Grebinski and Kucherov [GK00] extend this upper bound to any $W \leq n$. They gave a non-constructive non-adaptive algorithm that asks

$$\frac{4W \log\left(\frac{n}{W}+1\right)}{\log W}$$

[4] Again here we remind the reader that all the weighing complexities in this paper are multiplied by $1 \pm o(1)$

[5] In [D75] Djackov mention this bound without a proof.

weighings. Ruszinkó and Vanroose [RV97] gave the first adaptive polynomial time algorithm that solves the problem when $W = n$ in $O(W(\log\log W)/\log W)$ weighings.[6]

In this paper we give a polynomial time adaptive algorithm that makes

$$\frac{cW\log\left(\frac{n}{W}+1\right)}{\log W}$$

weighings where $c = 2$ for $W = o(n)$ and $c = 5$ for $W = O(n)$.

When $W > n$ the information theoretic lower bound for this problem gives

$$\frac{W\log\left(\frac{W}{n}+1\right)}{\log n},$$

when $W < n^2$ and n when $W > n^2$. So for $W > n^2$ the optimal algorithm is to weigh each one of the coins. When $n < W < n^2$, Ruszinkó and Vanroose [RV97] algorithm can be extended to any $W > n$ and is optimal for $W > n(\log n)^\alpha$ for any constant α. Grebinski and Kucherov [GK00] gave a non-constructive non-adaptive algorithm that makes

$$\frac{4n\log\left(\frac{W}{n}+1\right)}{\log n}$$

weighings.

In this paper we give an adaptive algorithm that finds all the weights in polynomial time in

$$\frac{cn\log\left(\frac{W}{n}+1\right)}{\log n}$$

weighings where $c = 2$ when $n = o(W)$ and 5 when $n = O(W)$.

To the best of our knowledge this is the first polynomial time optimal algorithm for this problem.

Our paper is organized as follows: In section 2 we give the new construction of search matrix and detecting matrix. In section 3 we give an optimal polynomial time algorithm for the d-coin weighing problem and in section 4 we give other coin weighing problems with a spring scale that can be solved using our new technique.

2 Search Matrix and Detecting Matrix

In this section we use Fourier representation to build optimal size search and detecting matrices.

2.1 Fourier Representation

Let $f(x_1,\ldots,x_\nu) : \{-1,+1\}^\nu \to \mathcal{R}$ be a real function. Define the basis

$$B = \left\{ \chi_a(x) = \prod_{a_i=1} x_i \,\middle|\, a \in \{0,1\}^\nu \right\}$$

for $\mathcal{R}^{\{-1,+1\}^\nu}$. It is known that B is orthonormal basis. Therefore, any function in $f \in \mathcal{R}^{\{-1,+1\}^\nu}$ can be uniquely represented as

$$f(x) = \sum_{a\in\{0,1\}^\nu} \hat{f}(a)\chi_a(x).$$

The coefficient $\hat{f}(a) \in \mathcal{R}$ is called the Fourier coefficient of χ_a and is equal to

$$\hat{f}(a) = \frac{1}{2^\nu}\sum_{x\in\{-1,+1\}} f(x)\chi_a(x).$$

Using the fast Fourier transform all the coefficients $\hat{f}(a)$ can be found from the values of $f(x)$, $x \in \{-1,+1\}$ and can be ordered according to the lexicographic order of $a \in \{0,1\}^\nu$ in time $O(\nu 2^\nu)$.

2.2 Search Matrix

A $k \times n$ (0,1)-matrix, M is called *search matrix* if for every $u, v \in \{0,1\}^n$ where $u \neq v$ we have $Mv \neq Mu$.

For $a \in \{0,1\}^\nu$ we denote by $|a|$ the Hamming weight of a. We say that $a \in S \subset \{0,1\}^\nu$ is *maximal element* in S if there is no $b \in S$ such that $b > a$ (in the usual lattice order).

Let $a \in \{0,1\}^\nu$ and suppose $a_{j_1}, a_{j_2}, \ldots, a_{j_{|a|}}$ are the entries of a that are one where $1 \le j_1 < j_2 < \cdots < j_{|a|} \le \nu$. For $1 \le k \le |a|$ we define the function

$$\hat{f}_{a,k}(x) = \left(2\prod_{i=1}^{k}\frac{x_{j_i}+1}{2} - 1\right) x_{j_{k+1}}x_{j_{k+2}}\cdots x_{j_{|a|}}$$

and

$$f_{a,k}(x) = \frac{\hat{f}_{a,k}(x)+1}{2}.$$

Let F_ν be the set of all functions $f_{a,k}(x)$, $1 \le k \le |a|$.

The following properties are easy to prove

Lemma 1 *We have*

1. $f_{a,k}(x) : \{-1,+1\}^\nu \to \{0,1\}$.
2. $|F_\nu| = \nu 2^{\nu-1}$.
3. *The Fourier coefficient of* χ_a *in* $f_{a,k}(x)$ *is* 2^{-k}.
4. *For any* $b \not< a$ *the Fourier coefficient of* χ_b *in* $f_{a,k}(x)$ *is* 0.

The following lemma is the key for our construction

Lemma 2 *Let* $S = \{(a^{(1)},k_1),\ldots,(a^{(\ell)},k_\ell)\} \subset \{0,1\}^\nu \times N$ *and*

$$f = \sum_{i=1}^{\ell} f_{a^{(i)},k_i}(x).$$

Then, the following are equivalent

1. $a^{(j_1)} = a^{(j_2)} = \cdots = a^{(j_t)} = a$ *and* $a^{(i)} \neq a$ *for all* $i \notin \{j_1, j_2, \ldots, j_t\}$ *and* a *is maximal element in* $S' = \{a^{(1)},\ldots,a^{(\ell)}\}$.

[6] Ruszinkó and Vanroose [RV97] claim, mistakenly, that Lindstrom algorithm runs in exponential time. Lindstrom algorithm runs in polynomial time and therefore their algorithm (that uses Lindstrom algorithm) runs in polynomial time.

2. *The Fourier coefficient of χ_a in f is $2^{-k_{j_1}} + 2^{-k_{j_2}} + \cdots + 2^{-k_{j_t}}$ and for every $b > a$ the Fourier coefficient of χ_b in f is 0.*

Proof. (1) $\Rightarrow$ (2) If a is maximal element in S' then for any element $b > a$ we have $b \not< c$ for all $c \in S'$. Therefore, by (4) in Lemma 1 the Fourier coefficient of χ_b in all $f_{a^{(i)},k_i}(x)$ is 0. This implies that the Fourier coefficient of χ_b in f is 0. Now since a is maximal element in S', for every $b \in S$, $b \neq a$ we have $a \not< b$. Therefore, the only functions that contribute to the coefficient χ_a are $f_{a^{(j_i)},k_{j_i}}(x)$, $i \leq t$. Now by (3) in Lemma 1 this coefficient is equal to $2^{-k_{j_1}} + 2^{-k_{j_2}} + \cdots + 2^{-k_{j_t}}$.

(2) $\Rightarrow$ (1) Suppose a is not maximal. Then there is $b > a$ that is maximal in S. By the above argument the coefficient of χ_b is not zero and we get a contradiction. Now since a is maximal, then as above, the only functions that contribute to the coefficient χ_a are all $f_{a^{(j_i)},k_{j_i}}(x)$ where $a^{(j_i)} = a$. By (3) in Lemma 1 the result follows. ■

Now to build the search matrix. Define M a $2^\nu \times \nu 2^{\nu-1}$ matrix where for every $x \in \{-1,1\}^\nu$ and $f_{a,k} \in F_\nu$, $M[x, f_{a,k}] = f_{a,k}(x)$. Here the matrix row are labeled with the elements of $\{-1,+1\}^\nu$ and the columns are labeled with the elements of F_ν. We now prove

Theorem 3 *We have*

1. *For any n there is a search matrix M of size $k \times n$ where*
$$k = \frac{2n}{\log n}.$$
2. *Given Mv for $v \in \{0,1\}^n$ the vector v can be found in time $O(n + d(n/\log n))$ where d is the weight of v.*

Proof Sketch. Consider the minimal ν such that $\nu 2^{\nu-1} \geq n$. Build the matrix M as above. Let $v \in \{0,1\}^{\nu 2^{\nu-1}}$. As we did for the columns of M, we will label the entries of v with $f_{a,\ell}$ and write $v_{f_{a,\ell}}$ for the value of this entry. The vector $Mv \in N^{2^\nu}$, though, will be labeled with $\{-1,1\}^\nu$. Notice that Mv gives a column vector which is the sum of the $f_{a,\ell}$th columns in M where $v_{f_{a,\ell}} = 1$. So this sum is equal to

$$f(x) = (Mv)[x] = \sum_{v_{f_{a,\ell}}=1} f_{a,\ell}(x)$$

for all $x \in \{-1,+1\}^\nu$. Using Fourier transform we can find all Fourier coefficients $\hat{f}(z)$ ordered in lexicographic order according to $z \in \{0,1\}^{2^\nu}$ in time $\nu 2^\nu = O(n)$. We search for a maximal z where $\hat{f}(z) \neq 0$. By Lemma 2, at least one $f_{a,r}$ can be detected. Then $v_{f_{a,r}} = 1$. We then recursively do the above for the Fourier representation of $f - f_{a,r}$.

This construction gives $k = 4n/\log n$. In the full paper we use a similar technique as in [L65, L75] to improve the bound by a factor of 2. ■

2.3 Detecting Matrix

A $(d_1, d_2, \ldots, d_n)$-detecting matrix M is a $(0,1)$-matrix such that for every $v, u \in \prod_i \{0, 1, \ldots, d_i - 1\}$, $v \neq u$ we have $Mv \neq Mu$. We extend the above construction to build an optimal $(d_1, d_2, \ldots, d_n)$-detecting matrix in polynomial time. To find v from Mv we use the same algorithm as in the previous subsection.

For $a, b \in \{0,1\}^\nu$, $b < a$ define the following function

$$f_{a,b}(x) = \prod_{a_i=1} \frac{(-1)^{b_i} x_i + 1}{2}.$$

Notice that $f_{a,b} : \{-1,+1\}^\nu \to \{0,1\}$ and the Fourier coefficient of χ_a in $f_{a,b}$ is $2^{-|a|}$ when $|b|$ is even and $-2^{-|a|}$ when $|b|$ is odd. Define

$$G_\nu = \{f_{a,b}(x) \mid b < a, |b| = even\}.$$

It is easy to see that $|G_\nu| = (3^\nu - 1)/2$.

We now prove

Theorem 4 *Let $1 < d_1 \leq d_2 \leq \cdots \leq d_n$ be integers and let ν be the maximal integer where*

$$(\nu - 2)2^{\nu-1} \leq \log \left(d_{n-2^\nu}^{2^\nu} \prod_{i=1}^{n-2^\nu} d_i \right).$$

There is a $(d_1, d_2, \cdots, d_n)$-detecting matrix of size $2^\nu \times n$.

Proof Sketch. We construct a $2^\nu \times n$ $(0,1)$-matrix M as follows. The matrix M will be $[M_1|M_2]$ where M_1 contains $n - 2^\nu$ columns and M_2 contains 2^ν columns. For each vector $a \in \{0,1\}^\nu$ we construct ℓ_a (specified below) columns in M_1 and one column in M_2. We now show the $\ell_a + 1$ columns that correspond to $a \in \{0,1\}^\nu$.

For any $a \in \{0,1\}^\nu$ suppose we have already constructed columns 1 to r in M_1 and columns 1 to s in M_2. Let ℓ_a be such that $d_{r+1} d_{r+2} \cdots d_{r+\ell_a} \leq 2^{|a|-1}$ and $d_{r+1} d_{r+2} \cdots d_{r+\ell_a+1} > 2^{|a|-1}$. Consider $G_a = \{f_{a,b}(x) : b < a, |b| = even\} \subset G_\nu$. We have $|G_a| = 2^{|a|-1}$. We take any subsets $G_{a,0}, G_{a,1}, \cdots, G_{a,\ell_a} \subseteq G_a$ where $|G_{a,0}| = 1$ and $|G_{a,i}| = d_{r+1} d_{r+2} \cdots d_{r+i}$ for $i = 1, 2, \ldots, \ell_a$. We construct ℓ_a columns that will be the functions[7] $g_{a,i} = \sum_{h \in G_{a,i}} h$ for $i = 0, \ldots, \ell_a - 1$. Those will be columns $r+1, r+2, \ldots, r+\ell_a$ of M_1 and therefore of M. Then we put the column $g_{a,\ell_a} = \sum_{h \in G_{\ell_a}} h$ in M_2. This will be column $s+1$ of M_2 and therefore it will be column $n - 2^\nu + s + 1$ of M.

It is easy to see that $g_{a,i} : \{-1,+1\}^\nu \to \{0,1\}$, the Fourier coefficient of χ_a in $g_{a,i}$ is $d_{r+1} d_{r+2} \cdots d_{r+i}/2^{|a|}$ and for every $b > a$, the Fourier coefficient of χ_b in $g_{a,i}$ is 0.

We do the above for all the vectors $a \in \{0,1\}^\nu \backslash \{\mathbf{0}\}$. Since

$$d_{r+1} d_{r+2} \cdots d_{r+\ell_a} d_{n-2^\nu} \geq$$
$$d_{r+1} d_{r+2} \cdots d_{r+\ell_a+1} \geq 2^{|a|-1}$$

we have

$$d_{n-2^\nu}^{2^\nu} \prod_{i=1}^{n-2^\nu} d_i \geq \prod_{a \in \{0,1\}^\nu \backslash \{\mathbf{0}\}} 2^{|a|-1} =$$

$$2^{(\nu-1)+\binom{\nu}{\nu-1}(\nu-2)+\binom{\nu}{\nu-2}(\nu-3)+\cdots+\binom{\nu}{2}1} \geq 2^{(\nu-2)2^{\nu-1}}$$

which implies the bound.

[7] As before, the rows are labeled with $\{-1,+1\}^\nu$. Row $x \in \{-1,+1\}^\nu$ in this column is equal to $g_{a,\ell_a}(x)$.

Now M is $(d_1, d_2, \cdots, d_n)$-detecting matrix follows from the fact that given $\lambda_{r+i} < d_{r+i}$ for $1 \leq i \leq \ell_a$ and any $\lambda_{r+\ell_a+1}$. Given $\lambda_{r+1} + \lambda_{r+2}d_{r+1} + \lambda_{r+3}d_{r+1}d_{r+2} + \cdots + \lambda_{r+\ell_a+1}d_{r+1}d_{r+2}\cdots d_{r+\ell_a}$ one can uniquely determines all λ_{r+i}. ■

Note: In the full paper we will use techniques from [L65, L75] and show that a $k \times n$, $(d_1, d_2, \cdots, d_n)$-detecting matrix exists where $(\log k - 4)(k/2) \leq \log\left(d_{n-k}^k \prod_{i=1}^{n-k} d_i\right)$ for any k (not only k that is a power of two). This improves the above bound by a factor of 2.

We now show

Corollary 5 *Let $1 < d_1 \leq d_2 \leq \cdots \leq d_n$ where $d_1 + d_2 + \ldots + d_n = d$. There is a $(d_1, d_2, \ldots, d_n)$-detecting matrix of size $k \times n$ where*

$$(\log k - 4)k \leq 2n \log \frac{d}{n}.$$

Proof. By Theorem 4 and using the fact that the geometric mean is less than or equal the arithmetic mean, there is a $(d_1, d_2, \ldots, d_n)$-detecting matrix of size $k \times n$ where

$$\frac{(\log k - 4)k}{2} \leq \log\left(d_{n-k}^k \prod_{i=1}^{n-k} d_i\right) \leq \log\left(\frac{d}{n}\right)^n = n \log \frac{d}{n}.$$

This implies the result. ■

3 The d-Coin Weighing Problem

In this section we give a polynomial time optimal adaptive algorithm for the d-coin weighing problem. As mentioned above the problem can be reduced to the following problem: Suppose we are given n coins out of a collection of coins of two distinct weights, 0 and 1, true and counterfeit coins, respectively, where d of them are counterfeit coins. Assume we are allowed to weigh subsets of coins in a spring scale. Determine the weight of each coin in a minimal number of weighing.

Let $v \in \{0,1\}^n$ be the vector of the weights of the coins. We will assume that n is a power of 2, otherwise take n the smallest power of 2 that is greater or equal to the number of coins. In the full paper we handle any n to get the best number of weighing possible. When we weigh coins $S = \{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$ we get the value $\delta(S) = v_{i_1} + \cdots + v_{i_k}$. Notice that for a partition $S = S_1 \cup S_2 \cup \cdots \cup S_t$ we have

$$\delta(S) = \delta(S_1) + \delta(S_2) + \cdots + \delta(S_t). \tag{2}$$

Our first (non-optimal) algorithm will define a sequence of $1 + \log n$ sets of disjoint sets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{\log n}$. The initial set is $\mathcal{S}_0 = \{\{1, 2, \ldots, n\}\}$. At stage ℓ, the algorithm takes each set $S \in \mathcal{S}_{\ell-1}$ splits it into two equal size disjoint sets $S = S_1 \cup S_2$ and weigh to find $\delta(S_1)$. By (2) $\delta(S_2) = \delta(S) - \delta(S_1)$. If $\delta(S_i) \neq 0$ then it adds S_i to $\mathcal{S}_{\ell-1}$. It is easy to see that the complexity of this algorithm is [UTW00]

$$d \log \frac{n}{d}.$$

This is not optimal for non-constant d.

Our (optimal) algorithm will exploit Corollary 5 to find all $\delta(S)$ for all $S \in \mathcal{S}_\ell$.

We run the above algorithm as long as the number of sets in $\mathcal{S}_\ell$ is less than $\sqrt{d}$. When $|\mathcal{S}_\ell| > \sqrt{d}$ let $\mathcal{S}_\ell = \{S_1, S_2, \ldots, S_q\}$ and $d_i = |S_i|$ for $i = 1, \ldots, q$. We split each set S_i into two equal size disjoint sets $S_i = S_{i,1} \cup S_{i,2}$. We construct a $(d_1 + 1, d_2 + 1, \ldots, d_q + 1)$-detecting matrix M of size $k \times q$. Then for every row i, M_i in M we weigh to find

$$\delta\left(\bigcup_{M_{i,j}=1} S_{j,1}\right) = \sum_{j=1}^{q} M_{i,j}\delta(S_{j,1}).$$

We then use the algorithm in the previous section to find all $\delta(S_{j,1})$. Then all $\delta(S_{j,2}) = d_j - \delta(S_{j,1})$ can be found. We put in $\mathcal{S}_{\ell+1}$ the nonempty sets $S_{j,1}, S_{j,2}$ and recursively run the above.

The reader may jump into conclusion that since we save a factor of $\log d$ weighing at each stage the complexity will be $(d/\log d) \log n/d$. Well, this is not quite true. When $|\mathcal{S}_\ell| = \sqrt{d}$ the number of weighing is $O(\sqrt{d})$, so no saving achieved at this level.

In the next section we give some inequalities that will be used in the analysis of the algorithm and then we give the analysis of the algorithm

3.1 Some Inequalities

In this section we give some inequalities that will be used in the paper.

We first give the following trivial inequalities

Lemma 6 *1. For any $\sigma > 1$ and $0 < x \leq \frac{\sigma-1}{\sigma}$ we have*

$$1 + x < \frac{1}{1-x} \leq 1 + x + \sigma x^2.$$

We now would like to bound k when $k \log k \leq x$ where $x > 1$. The following simple bound

$$k \leq \frac{2x}{\log x} \tag{3}$$

can be derived as follows: Let k_0 be a real number such that $k_0 \log k_0 = x$. Then $k \leq k_0$. Since $k_0^2 \geq k_0 \log k_0 = x$ we have $k_0 \geq x^{1/2}$ and

$$k \leq k_0 = \frac{x}{\log k_0} \leq \frac{2x}{\log x}.$$

To get a better bound we prove the following

Lemma 7 *Let $x \geq 8$ be a real number. If k is an integer and*

$$k \log k \leq x$$

then

$$k \leq \frac{x}{\log x}\left(1 + \frac{\log\log x}{\log x} + \frac{c}{\log x}\right)$$
$$= \frac{x}{\log x}\left(1 + O\left(\frac{\log\log x}{\log x}\right)\right),$$

for some constant $c < 5$.

3.2 Analysis of the Algorithm

In this subsection we show that the weighing complexity of the algorithm is

$$\frac{2d\log\frac{n}{d}}{\log d}+O\left(\frac{d}{\log d}+\frac{d(\log\log d)\log\frac{n}{d}}{(\log d)^2}\right).$$

This complexity is

$$\frac{2d\log\frac{n}{d}}{\log d}\left(1+O\left(\frac{\log\log d}{\log d}\right)\right)$$

for $d=o(n)$.

We first prove

Lemma 8 *For $\ell\in\mathcal{R}^+$,*

$$B_\ell=\begin{cases}(\ell-4)2^\ell & \ell\le\frac{\log d}{2}\\ 2^{\ell+1}\log\left(\frac{d}{2^\ell}+1\right) & \frac{\log d}{2}<\ell\le\log d\\ 2d & \ell>\log d\end{cases}.$$

The number of weighings C_ℓ at level ℓ of the algorithm satisfies:

$$(\log C_\ell-4)C_\ell\le B_\ell.$$

Proof. First, it is easy to see that B_ℓ is monotone non-decreasing function in ℓ.

Since the algorithm splits the sets into two almost equal size disjoint sets the size of the sets in $\mathcal{S}_\ell$ is at most $\lceil n/2^\ell\rceil$ and the number of sets in $\mathcal{S}_\ell$ is at most 2^ℓ.

Suppose $\ell\le(\log d)/2$. Since for $\ell\le(\log d)/2$, $2^\ell\le\sqrt{d}$ the algorithm makes at most 2^ℓ weighings at stage $\ell\le(\log d)/2$ and therefore $(\log C_\ell-4)C_\ell\le(\ell-4)2^\ell=B_\ell$ for $\ell\le(\log d)/2$.

At stages $(\log d)/2<\ell\le\log d$ we have two cases. If the number of sets in $\mathcal{S}_\ell$ is less that $\sqrt{d}$ then the number of weighings C_ℓ is at most $\sqrt{d}$ and since B_ℓ is monotone non-decreasing function we have

$$(\log C_\ell-4)C_\ell\le\left(\frac{\log d}{2}-4\right)2^{\frac{\log d}{2}}=B_{\frac{\log d}{2}}\le B_\ell.$$

The other case is when the number of sets n in $\mathcal{S}_\ell$ is greater than $\sqrt{d}$ then since $\mathcal{S}_\ell$ contains at most 2^ℓ sets, by Corollary 5 the number of weighings is C_ℓ where

$$\begin{aligned}(\log C_\ell-4)C_\ell &\le 2n\log\left(\frac{d+n}{n}\right)\\ &\le 2^{\ell+1}\log\left(\frac{d+2^\ell}{2^\ell}\right)\\ &= 2^{\ell+1}\log\left(\frac{d}{2^\ell}+1\right)=B_\ell.\end{aligned}$$

Now when $\ell>\log d$ then, as above, since B_ℓ is monotone non-decreasing function in ℓ it is enough to consider the case where $|\mathcal{S}_\ell|\ge\sqrt{d}$. Since the number of counterfeit coins is bounded by d we have $n=|\mathcal{S}_\ell|\le d$ and by Corollary 5 the number of weighings is at most C_ℓ where

$$\begin{aligned}(\log C_\ell-4)C_\ell &\le 2n\log\left(\frac{d+n}{d}\right)\\ &\le 2d\log\left(\frac{d+d}{d}\right)=2d.\end{aligned}$$

This completes the proof. ■

We now count the total number of weighings.

For $\ell\le(\log d)/2$ we have $C_\ell\le 2^\ell$ and therefore the total number of weighings in the first $\lfloor(\log d)/2\rfloor$ stages is at most

$$\sum_{\ell=1}^{\lfloor\frac{\log d}{2}\rfloor}2^\ell\le 2^{\frac{\log d}{2}+1}=2\sqrt{d}. \tag{4}$$

For $\frac{\log d}{2}<\ell\le\log d$ we have $(\log C_\ell-4)C_\ell\le B_\ell$. Since $(C_\ell/16)\log(C_\ell/16)\le(B_\ell/16)$ and $B_{\frac{\log d}{2}}\ge\sqrt{d}$, by (3)

$$C_\ell\le\frac{2B_\ell}{\log(B_\ell/16)}\le\frac{2B_\ell}{\log B_{\frac{\log d}{2}}-4}\le\frac{4B_\ell}{\log d-8}. \tag{5}$$

Now

$$\begin{aligned}\sum_{\ell=\lceil(\log d)/2\rceil}^{\lfloor\log d\rfloor}B_\ell &= \sum_{\ell=\lceil(\log d)/2\rceil}^{\lfloor\log d\rfloor}2^{\ell+1}\log\left(\frac{d}{2^\ell}+1\right)\\ &\le \sum_{\substack{\ell=\log d,\log d-1\\ ,\dots,\\ \log d-\lceil(\log d)/2\rceil}}2^{\ell+1}\log\left(\frac{d}{2^\ell}+1\right)\\ &\le \sum_{i=0}^{\lceil(\log d)/2\rceil}d\frac{\log(2^i+1)}{2^{i-1}}\\ &\le d\sum_{i=0}^{\infty}\frac{\log(2^i+1)}{2^{i-1}}\\ &\le 8d.\end{aligned}$$

Therefore, by (5),

$$\begin{aligned}\sum_{\ell=\lceil(\log d)/2\rceil}^{\lfloor\log d\rfloor}C_\ell &\le \frac{4\sum_{\ell=\lceil(\log d)/2\rceil}^{\lfloor\log d\rfloor}B_\ell}{\log d-8}\\ &\le \frac{32d}{\log d-8}\\ &= \frac{32d}{\log d}+O\left(\frac{d}{(\log d)^2}\right).\end{aligned} \tag{6}$$

Now for $\ell>\log d$ we have $(\log C_\ell-4)C_\ell\le 2d$ and therefore by Lemma 7

$$C_\ell\le\frac{2d}{\log d}+O\left(\frac{d\log\log d}{(\log d)^2}\right).$$

Since the number of stages is $\lceil\log n\rceil$ we get

$$\sum_{\ell=\lceil\log d\rceil}^{\lceil\log n\rceil}C_\ell \le \frac{2d\log\frac{n}{d}}{\log d}+ \tag{7}$$

$$\frac{2d}{\log d}+O\left(\frac{d(\log\log d)\log\frac{n}{d}}{(\log d)^2}\right). \tag{8}$$

By (4), (6) and (7) the result follows.

4 A General Coin Weighing Problems

In this section we give an optimal polynomial time algorithm for the following general coin weighing problem:

Suppose we are given n coins out of a collection of coins of distinct unknown non-negative integer weights $w_1, w_2, \cdots, w_n$ where

$$\sum_{i=1}^{n} w_i = W.$$

Assume we are allowed to weigh subsets of coins in a spring scale. Determine the weight of each coin in a minimal number of weighings.

An information-theoretic argument gives the lower bound

$$k(n,W) = \begin{cases} \frac{W\log\left(\frac{n}{W}+1\right)}{\log W} & W \le n \\ \frac{n\log\left(\frac{W}{n}+1\right)}{\log n} & n < W \le n^2 \\ n & W > n^2 \end{cases} \quad (9)$$

This problem was studied for $W \le n$ in [P81, RV97, GK00] and for $W > n$ in [GK00]. Pippenger [P81] showed nonconstructively that for $W = n$ there is a non-adaptive algorithm that makes $O(W/\log W)$ weighings. Grebinski and Kucherov [GK00] extend this upper bound to any $W \le n$. They gave a non-constructive non-adaptive algorithm that asks

$$\frac{4W\log\left(\frac{n}{W}+1\right)}{\log W}$$

weighings. Ruszinkó and Vanroose [RV97] gave the first adaptive polynomial time algorithm that solves the problem when $W = n$ in $O(W(\log\log W)/\log W)$ weighings.

For $W \le n$ we give a polynomial time adaptive algorithm that makes

$$\frac{2W\log\left(\frac{n}{W}+1\right)}{\log W}$$

weighings when $W = o(n)$ and

$$\frac{c_\beta W\log\left(\frac{n}{W}+1\right)}{\log W}$$

weighings when $W = \beta n$ where for any constant β

$$c_\beta \le c_1 = \sum_{j=1}^{\infty} \frac{\log(2^j+1)}{2^{j-1}} = 4.803048.$$

See the table at the end of this section for diferent values of c_β.

When $W > n$ the information theoretic lower bound for this problem gives

$$\frac{W\log\left(\frac{W}{n}+1\right)}{\log n},$$

when $W < n^2$ and n when $W > n^2$. So for $W > n^2$ the optimal algorithm is to weigh each one of the coins. When $n < W < n^2$, Ruszinkó and Vanroose [RV97] algorithm can be extended to any $W > n$ and is optimal for $W > n(\log n)^\alpha$ for any constant α. Grebinski and Kucherov [GK00] gave a non-constructive non-adaptive algorithm that makes

$$\frac{4n\log\left(\frac{W}{n}+1\right)}{\log n}$$

weighings.

Here we give an adaptive algorithm that finds all the weights in polynomial time in

$$\frac{2n\log\left(\frac{W}{n}+1\right)}{\log n}$$

weighings for $W = \omega(n)$ and

$$\frac{c'_\beta n\log\left(\frac{W}{n}+1\right)}{\log n}$$

weighings for $W = \beta n$ where $c'_\beta < c_1 = 4.803048$.

4.1 The Algorithm

Let $w \in \mathcal{N}^n$ be the vector of the weights of the coins where $\mathcal{N}$ is the set of non-negative integers where

$$\sum_{i=1}^{n} w_i = W.$$

When we weigh coins $S = \{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$ we get the value $w(S) = w_{i_1} + \cdots + w_{i_k}$.

The algorithm is similar to the one in section 3.

We start with an initial set $\mathcal{S}_0 = \{S_1, S_2, \ldots, S_q\}$ of ℓ disjoint subsets of $\{1, 2, \ldots, n\}$ where

1. For every $1 \le i, j \le q$, $||S_i| - |S_j|| \le 1$.
2. $S_1 \cup S_2 \cup \cdots \cup S_q = \{1, 2, \ldots, n\}$.

The constant q depends on n and will be determined later in the analysis of the algorithm.

We find $w(S_i)$ for all $i = 1, \ldots, q$ and then run the non-optimal algorithm in section 3 as long as the number of sets in $\mathcal{S}_\ell$ is less than $d/\log^3 d$ where $d = \min(n, W)$. When $|\mathcal{S}_\ell| > d/\log^3 d$ let $\mathcal{S}_\ell = \{S_1, S_2, \ldots, S_q\}$ and $w_i = w(S_i)$ for $i = 1, \ldots, q$. We split each set S_i into two almost equal size disjoint sets $S_i = S_{i,1} \cup S_{i,2}$. We construct a $(w_1 + 1, w_2 + 1, \ldots, w_q + 1)$-detecting matrix M of size $k \times q$. Then for every row i, M_i in M we weigh to find

$$w\left(\bigcup_{M_{i,j}=1} S_{j,1}\right) = \sum_{j=1}^{q} M_{i,j} w(S_{j,1}).$$

We then use the algorithm in section 3 to find all $w(S_{j,1})$. Then all $w(S_{j,2}) = w_j - w(S_{j,1})$ can be found. We put in $\mathcal{S}_{\ell+1}$ the nonempty sets $S_{j,1}, S_{j,2}$ and recursively run the above.

For the analysis of the algorithm we will consider the two case where $W \ge n$ and $n \ge W$. In this abstract we will only give the analysis for the case $W \ge n$. The other case is very similar and will be given in the full paper.

4.2 Analysis of the Algorithm when $W \ge n$

In this subsection we give the analysis of the algorithm when $W \ge n$. We will first assume that $q = 1$ and then show how to choose q to achieve a better weighing complexity.

Consider a binary tree $\mathcal{T}$ where each of its nodes is labeled with a subset of $S_0 = \{1, 2, \ldots, n\}$. The root of the tree is labeled with $S_0 = \{1, 2, \ldots, n\}$ and when a node is

labeled with $S_i = \{j, j+1, \ldots, j+p\}$, then this node is a leaf if $p = 0$ and, otherwise, it is an internal node with two children where the left child is labeled with $S_{2i+1} = \{j, j+1, \ldots, j+\lfloor p/2 \rfloor\}$ and the right child is labeled with $S_{2i+2} = \{j+\lfloor p/2 \rfloor + 1, \ldots, p\}$.

We can regard algorithm $\mathcal{A}$ in the previous section (with $q = 1$) as an algorithm that at stage ℓ finds the weight $w(S_i)$ for all S_i at level ℓ in the tree $\mathcal{T}$. The above algorithm do not consider a node that is labeled with S_i when $w(S_i) = 0$. We change the above algorithm to an algorithm $\mathcal{A}'$ that considers also those nodes. Obviously, the complexity of $\mathcal{A}'$ is greater or equal to the complexity of $\mathcal{A}$.

Let $n = 2^t + n'$ where $n' < 2^t$. Then the number of nodes in levels $\ell = 1, 2, 3, \ldots, t+2$ of the tree $\mathcal{T}$ are $d_\ell = 1, 2, 2^2, \cdots, 2^t, 2n'$, respectively. Algorithm $\mathcal{A}'$ finds the weights of half of the sets in each level. Therefore, by Corollary 5, the number of weighings for level ℓ is k_ℓ where $k_\ell = d_\ell/2$ if $d_\ell < n/\log^3 n$ and

$$k_\ell(\log k_\ell - 4) \le d_\ell \log\left(\frac{2W}{d_\ell} + 1\right),$$

otherwise. Let

$$W = \beta n$$

where $1 \le \beta \le n$. Let $2^t = \alpha n$ where $1 \ge \alpha > 1/2$. Then $n' = n - 2^t = (1-\alpha)n$.

If $d_{t+2} = 2n' < 2n/\log^3 n$ then $k_{t+2} = n' < n/\log^3 n$. Otherwise, by Lemma 7,

$$\begin{aligned} k_{t+2} &\le \frac{2n' \log\left(\frac{2W}{n'} + 1\right)}{\log\left(2n' \log\left(\frac{2W}{n'} + 1\right)\right)}(1 + o(1)) \\ &= 2(1-\alpha)\log\left(\frac{\beta}{1-\alpha} + 1\right)\frac{n}{\log n}(1 + o(1)) \\ &= 2\bar{\alpha}\log\left(\frac{\beta}{\bar{\alpha}} + 1\right)\frac{n}{\log n}(1 + o(1)), \end{aligned}$$

where $\bar{\alpha} = 1 - \alpha$.

Now at level $t+2-j$, $j = 1, 2, \ldots$ we have if $d_{t+2-j} < 2n/\log^3 n$ then $k_{t+2-j} < n/\log^3 n$. Otherwise, by Lemma 7

$$\begin{aligned} k_{t+2-j} &\le \frac{d_{t+2-j}\log\left(\frac{2W}{d_{t+2-j}} + 1\right)}{\log\left(d_{t+2-j}\log\left(\frac{2W}{d_{t+2-j}} + 1\right)\right)}(1 + o(1)) \\ &= \frac{2^{t+1-j}\log\left(\frac{W}{2^{t-j}} + 1\right)}{\log n}(1 + o(1)) \\ &= \frac{\alpha}{2^{j-1}}\log\left(\frac{2^j\beta}{\alpha} + 1\right)\frac{n}{\log n}(1 + o(1)). \end{aligned}$$

Therefore in the worst case the number of weighings is

$$\begin{aligned} \sum_{i=1}^{t+2} k_i &\le \left(2\bar{\alpha}\log\left(\frac{\beta}{\bar{\alpha}} + 1\right) + \sum_{j=1}^{\infty}\frac{\alpha}{2^{j-1}}\log\left(\frac{2^j\beta}{\alpha} + 1\right)\right) \\ &\quad \times\frac{n}{\log n}(1 + o(1)) \\ &= \frac{\left(2\bar{\alpha}\log\left(\frac{\beta}{\bar{\alpha}} + 1\right) + \sum_{j=1}^{\infty}\frac{\alpha}{2^{j-1}}\log\left(\frac{2^j\beta}{\alpha} + 1\right)\right)}{\log(\beta+1)} \\ &\quad \times\frac{n}{\log n}\log\left(\frac{W}{n} + 1\right)(1 + o(1)). \end{aligned}$$

It is easy to see that

$$\begin{aligned} \phi(\alpha, \beta) &= \frac{\left(2\bar{\alpha}\log\left(\frac{\beta}{\bar{\alpha}} + 1\right) + \sum_{j=1}^{\infty}\frac{\alpha}{2^{j-1}}\log\left(\frac{2^j\beta}{\alpha} + 1\right)\right)}{\log(\beta+1)} \\ &\to 2 \end{aligned}$$

when $\beta \to \infty$ and therefore for $W = \omega(n)$ the bound is

$$\frac{2n}{\log n}\log\left(\frac{W}{n} + 1\right)(1 + o(1)).$$

This bound give the worst case constant

$$\phi(0.825, 1) = 5.28133.$$

We will now show how to get rid of the first term in ϕ and make $\alpha = 1$ and then the constant will be

$$\phi(1, \beta) = \frac{\sum_{j=1}^{\infty}\frac{1}{2^{j-1}}\log\left(2^j\beta + 1\right)}{\log(\beta+1)}.$$

The idea is to choose $q = \lfloor n/2^k \rfloor$ where $k = \lceil 3\log\log n \rceil$. Then the first set $\mathcal{S}_0$ contains

$$q = \lfloor n/2^k \rfloor = O\left(\frac{n}{\log^3 n}\right).$$

Now we can write

$$n = 2^k \lfloor n/2^k \rfloor + n'$$

where

$$n' < 2^k = 2^{\lceil 3\log\log n \rceil} \le 2\log^3 n.$$

Then the number nodes in levels $\ell = 1, 2, 3, \ldots, k+2$ of the tree $\mathcal{T}$ are $d_\ell = 1, q, 2q, 2^2q, \cdots, 2^kq, 2n'$, respectively. Therefore for this case the first term will go into the $o(1)$ of the complexity and using the same analysis as above we get the bound on the number of weighings.

The following table show diferent values of c_β

β	c
1	4.803048
1.5	4.338263
2	4.060772
2.143311	4.000000
13.56659	3.000000
100	2.597891
1000000	2.200687
$\omega(1)$	2

Acknowledgment. I would like to thank Samer Banna for helping me find the relevant papers in this area.

References

[A88] M. Aigner. Combinatorial search. John Wiley and Sons, (1988).

[A86] M. Aigner. Search problems on graphs. *Discrete Applied Mathematics*, V. 14 , I. 3, pp. 215 - 230 (1986).

[AS85] M. Aigner and M. Schughart. Determining defectives in a linear order. *J. Statist. Plan. Inform.* 12 pp. 359-368 (1985).

[BLM08] N. H. Bshouty, P. Long and H. Mazzawi. Reconstructing graph from edge counting query. In preparation. (2008).

[C62] D. Cantor. Determining a set from the cardinalities of its intersections with other sets, *Canadian Journal of Mathematics*, V. 16, pp. 94-97. (1962)

[C79] J. Capetanakis. Tree algorithms for packet broadcast channels *Information Theory, IEEE Transactions on*, V. 25, I. 5 pp. 505- 515, (1979).

[Ca79] J. Capetanakis. Generalized TDMA: The Multi-Accessing Tree Protocol *Communications, IEEE Transactions*, V. 27, I. 10, Part 1 pp. 1476-1484. (1979).

[C80] C. Christen. A Fibonaccian algorithm for the detection of two elements. Publ. 341, Dept. d'IRO, Univ. Montreal. (1980)

[C83] C. Christen. Optimal detection of two complementary coins. *SIAM J. Algebra Discrete Methods*, 4, pp. 101-110, (1983).

[CH84] C. Christen and F. K. Hwang. Detection of a defective coin with partial weight information, *American Mathematical Monthly*, 91, pp. 173-179. (1984).

[CK08] S. Choi , J. H. Kim. Optimal query complexity bounds for finding graphs. Proceedings of the 40th annual ACM symposium on Theory of computing, STOC 08, pp. 749-758, (2008).

[CM66] D. G. Cantor and W. H. Mills. Determining a Subset from Certain Combinatorial Properties. *Canad. J. Math*, V. 18, pp. 42-48, (1966).

[CLZ01] G. Cohen, S. Litsyn and Zémor. Binary B2-Sequences: A New Upper Bound. *Journal of Combinatorial Theory*, Series A, V. 94, N. 1, pp. 152-155, (2001).

[CW79] S. Chang and E. J. Weldon. Coding for T-user multiple-access channel. *IEEE Tran. on inf. theo.*, V. 25, N. 6, november, (1979).

[D75] A. G. Djackov. On a search model of false coins. In Topics in Information Theory (Colloquia Mathematica Societatis Janos Bolyai 16, Keszthely, Hungary). Budapest, Hungary: Hungarian Acad. Sci., pp. 163-170, (1975).

[DH93] D. Du and F. K. Hwang. Combinatorial group testing and its application, V. 3 of Series on applied mathematics. World Science, (1993).

[ER63] Erdös and A. Rényi, On two problems of information theory. *Publ. Math. Inst. Hung. Acad. Sci.*, V. 8, pp. 241-254, (1963).

[F60] N. I. Fine, Solution of problem E 1399, *American Mathematical Monthly*, **67** p. 697, (1960).

[G98] V. Grebinski, On the power of additive combinatorial search model. In Proceedings of the Fourth Annual International Computing and Combinatorics Conference (COCOON'98), V. 1449 of LNCS, pp. 194-203. (1998).

[GK00] V. Grebinski and G. Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1), pp. 104-124 (2000).

[H90] F. Hao. The optimal procedures for quantitative group testing. *Discrete Applied Mathematics*, 26, pp. 79-86, (1990).

[H99] T. Hofmeister. An application of codes to attribute-efficient learning Source Lecture Notes In Computer Science; Vol. 1572 archive Proceedings of the 4th European Conference on Computational Learning Theory, pp. 101-110, (1999).

[K76] D. Knuth. The Computer as a Master Mind, *Journal of Recreational Mathematics* (9): 1-6, (1976-77)

[KLT00] G Kabatianski, V. Lebedev, J.Thorpe. The Mastermind game and the rigidity of the Hamming space. Proceedings. IEEE International Symposium on Information Theory, p. 375, (2000).

[L64] B. Lindström, On a combinatory detection problem I, *Mathematical Institute of the Hungarian Academy of Science*, 9, pp. 195207, (1964).

[L65] B. Lindström. On a combinatorial problem in number theory. *Canad. Math. Bull.*, 8: pp. 477-490, (1965).

[L66] B. Lindström, On a combinatory detection problem II, *Studia Scientiarum Mathematicarum Hungarica*, 1, pp. 353-361, (1966).

[L71] B. Lindström. On Möbius functions and a problem in combinatorial number theory. *Canad. Math. Bull.*, 14(4), pp. 513-516, (1971).

[L72] B. Lindström. On B_2-sequences of vectors, *J. Number Theory*, 4, pp. 261-265 (1972).

[L75] B. Lindström. Determining subsets by unramified experiments. In J.N. Srivastava, editor, *A Survey of Statistical Designs and Linear Models*, North Holland, Amsterdam, pp. 407-418. (1975).

[LV91] M. Li, P. M. B. Vitányi. Combinatorics and Kolmogorov Complexity. Structure in Complexity Theory Conference. pp. 154-163 (1991).

[M70] L. Moser. The second moment method in combinatorial analysis. In. Combinatorial Structures and their applications, Gordon and Breach, pp. 283-384, (1970).

[M81] J. L. Massey. Collision-resolution algorithms and random-access communications. In G. Longo, editor,Multi-User Communications Systems, pp. 73137, (1981).

[MK87] S. S. Martirosyan, G. G. Khachatryan. Construction of signature codes and the coin weighing problem. *Probl. Peredachi Inf.* 25: 4, 96-97 and Problrms of Information Transmission, 25: 4, 334-335 (1989).

[P77] N. Pippenger. An Information-Theoretic Method in Combinatorial Theory. *J. Comb. Theory, Ser. A*, 23(1), 99-104 (1977).

[P81] N. Pippenger. Bounds on the performance of protocols for a multiple-access broadcast channel. *IEEE Transactions on Information Theory*, 27(2), pp. 145-151, (1981).

[RV97] M. Ruszinkó, P. Vanroose. How an Erdös-

Rényi-type search approach gives an explicit code construction of rate 1 for random access with multiplicity feedback. *IEEE Transactions on Information Theory*, 43(1), pp. 368-372 (1997).

[S60] H. S. Shapiro, Problem E 1399. *American Mathematical Monthly*, **67**, p. 82, (1960).

[S38] J. Singer. A theorem in finite projective geometry and some applications to number theory. *Trans. Amer. Math. Soc.* 43, pp. 377-385, (1938).

[SS63] S. Soderberg, H. S. Shapiro. A combinatory detection problem. *American Mathematical Monthly*, **70**, pp. 1066-1070, (1963).

[TM78] B. Tsybakov, V. Mikhailov. Free Synchronous Packet Access in a Broadcast Channel with Feedback, *Problemy Peredachi Informassi*, 14(4), 259-280. (1978).

[UTW00] R. Uehara, K. Tsuchida and I. Wegener. Identification of partial disjunction, parity, and threshold functions. Theoretical Computer Science, V. 230, I. 1-2, pp. 131-147, (2000)