ELSEVIER Discrete Mathematics 163 (1997) 173–200

DISCRETE MATHEMATICS

A ($\log_2 3 + \frac{1}{2}$)-competitive algorithm for the counterfeit coin problem[1]

Peng-Jun Wan, Ding-Zhu Du*

Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, USA

Received 2 February 1995; revised 14 July 1995

Abstract

Consider a set of coins where each coin is either of the heavy type or the light type. The problem is to identify the type of each coin with minimal number of weighings on a balanced scale. The case that only one coin, called a counterfeit, has a different weight from others, is a classic mathematical puzzle. Later works study the case of more than one counterfeit, but the number of counterfeits is always assumed known. Recently, Hu and Hwang gave an algorithm which does not depend on the knowledge of the number of counterfeits, and yet perform uniformly good whatever that number turns out to be in the sample considered. Such an algorithm is known as a competitive algorithm and the uniform guarantee is measured by its competitive constant. Their algorithm has competitive ratio $2\log_2 3$. In this paper, we give a new competitive algorithm with competitive ratio $\log_2 3 + \frac{1}{2}$.

1. Introduction

Consider a set of n coins which contains d light coins and $n-d$ heavy coins (the cases, d is known or unknown are considered as two different models). We want to sort the coins by using a balance scale and call a test every time of using the balance scale. The problem is how to arrange the tests in order to identify the d light coins and $n-d$ heavy coins by using minimum number of tests.

Let $M_A(n:d)(M_A(n,d))$ denote the maximum number of tests required by an algorithm A to sort a (n,d) problem when d is unknown (known) before testing. Let

$$M(n:d) = \min_A M_A(n:d)$$

$$(M(n,d) = \min_A M_A(n,d))$$

* Corresponding author. E-mail: dzd@cs.umn.edu.

[1] Support in part by the National Science Foundation under grant CCR-9208913.

SSDI 0012-365X(95)00314-2

An algorithm A is called a c-competitive algorithm if there exits a constant b such that for all $0 < d < n$,

$$M_A(n:d) \leqslant c \cdot M_A(n, d) + b$$

and c is called the competitive ratio.

Hu and Hwang [5] first proposed a bisecting algorithm with a competitive ratio $3\log_2 3$. Soon after, Hu et al. [4] discovered a doubling algorithm with competitive ratio $2\log_2 3$. In this paper, we present a new doubling–backtracking algorithm with competitive ratio $\log_2 3 + \frac{1}{2}$.

2. Preliminaries

The analysis of competitive ratio involves both lower-bound and upper-bound problems. In this section, we first list some results about the lower-bound for $M_A(n:d)$ and $M_A(n, d)$.

Hu and Hwang [5] gave a lower-bound for $M_A(n, d)$:

Lemma 2.1.

$$M_A(n, d) \geqslant \frac{d}{\log_2 3}\left(\log_2 \frac{n}{d} + \log_2 \frac{e\sqrt{3}}{2}\right) - \frac{\log_2 d}{2\log_2 3} - \frac{0.567}{\log_2 3} - \frac{1}{2}.$$

Cairns [1] discovered an optimal algorithm to find the single counterfeit in a set of coins, when it is known before testing there is only one counterfeit.

Lemma 2.2. $M(n, 1) = \lceil \log_3 n \rceil$.

Hu and Hwang [3] also gave the value of $M(n:1)$.

Lemma 2.3. $M(n:1) = \lceil \log_2 n \rceil$.

For convenience we assume that the value of function $d\log_2 \frac{n}{d}$ at $d = 0$ is 0 because $\lim_{d\to 0} d\log_2 \frac{n}{d} = 0$. The following lemma, given by Du and Park [3], is an important tool for analysis [2].

Lemma 2.4. *Let $d = d_1 + d_2$ and $n = n_1 + n_2$ where $d_1 \geqslant 0, d_2 \geqslant 0, n_1 > 0$ and $n_2 > 0$. Then*

$$d_1 \log_2 \frac{n_1}{d_1} + d_2 \log_2 \frac{n_2}{d_2} \leqslant d \log_2 \frac{n}{d}.$$

3. Algorithms

A set of coins are called *uniform* if they are all of the same type, and called *unique* if there is only one exception. We also use the modifier '*heavy*' and '*light*' to specify the type of the majority of coins in a uniform or unique set. For a set of S of coins, $|S|$ denotes the cardinality of S and $\|S\|$ denotes the total weight of coins in S. Let X and Y be two nonempty sets of coins, then a comparison between X and Y means to compare $X' \subseteq X$ with $Y' \subseteq Y$ such that $|X'| = |Y'|$ and either $X' = X$ or $Y' = Y$. In other words, we compare two largest equinumerious subsets of X and Y. A comparison can have three possible outcomes: $\|X'\| = \|Y'\|$, $\|X'\| > \|Y'\|$, $\|X'\| < \|Y'\|$. We say the comparison yields equality for the first outcome, and yields inequality for the other two inequalities. A comparison path of a procedure is the series of comparison outcomes in the testing order.

The idea of our doubling–backtracking algorithm is as follows: The algorithm compares two sets of size 2^i ($i = 0, 1, 2, \ldots$). If the comparison yields equality, we get a uniform set of size 2^{i+1} by merging the two sets and then we fetch a disjoint set of size 2^{i+1} to compare the two sets of size 2^{i+1}. If the comparison yields inequality, we can identify the type of the previous uniform set of size 2^i. We then use a bisecting algorithm on the other set of size 2^i to find at least one coin of different type. We then backtrack the comparison path for the bisecting procedure to explore as much information as possible from the comparison path. After the backtracking, we either end our algorithm, or start another cycle of doubling, or continue the doubling comparison of size 2^i (*not* 2^{i+1}) until the next inequality yield appears, which is dealt in the same way.

We first describe some variables used in the algorithm:

S: the input set of coins
U: all the coins of unknown type (initially is S)
L: all identified light coins (initially is empty)
H: all identified heavy coins (initially is empty)
Continue: a boolean variable indicating the actions aftr a backtracking (continue one main loop or restart another main loop).
π: a linked list representing the comparison path for a bisecting procedure. Each element is of form $[X'_1 ? X_1]$, where ? could be $>$, $=$ or $<$, which corresponds to a result of comparison between X'_1 and X_1. Two neighbouring elements, say $\ldots [X'_1 ? X_1] \to [X'_2 ? X_2] \ldots$, have the relation $X_1 = X'_2 \cup X_2$.

We now describe some procedures that will be used by our algorithm.

3.1. UNIQUE-L and UNIQUE-H

The procedure UNIQUE-L (UNIQUE-H) takes as input a light (heavy) unique set of coins and use the algorithm given by Cairns [1] to identify the types of all coins in the input. We only give the code for UNIQUE-L here. The code for UNIQUE-H is similar.

Procedure UNIQUE-L(X);
  Use the algorithm of Cairns [1] on X and let x be the light coin;
  $L := L \cup \{x\}$;
  $H := H \cup (X - \{x\})$;
  $U := U - X$;
end-procedure

3.2. DIG-L, DIG-H

The procedure DIG-L (DIG-H) takes as input the set X containing at least one light (heavy) coin and use the bisecting method to find one light (heavy) coin. Here also we give the code only for DIG-L. The code for DIG-H is similar. Notice that in DIG-L, when $|X'| < |X''|$ and $\|X' \cup \{h\}\| = \|X''\|$, next time we test on X'' rather than on X'. This choice has great importance in our analysis.

Procedure DIG-L(X);
  $\pi := \emptyset$;
  repeat
    $X' := \left\lfloor \frac{|X|}{2} \right\rfloor$ coins from X;
    $X'' := X - |X'$;
    if $|X'| < |X''|$
    then pick up a heavy coin h and compare $X' \cup \{h\}$ with X'';
    else compare X' to X'';
    if $(|X'| = |X''|$ and $\|X'\| < \|X''\|)$ or $(|X'| < |X''|$ and $\|X' \cup \{h\}\| < \|X''\|)$
    then $X := X'$;
      $\pi := \pi \to [X'' > X']$;
    else if $(|X'| = |X''|$ and $\|X'\| = \|X''\|)$ or $(|X'| = |X''|$ and $\|X' \cup \{h\}\| < \|X''\|)$
      then $X := X''$;
        $\pi := \pi \to [X' = X'']$;
      else $X := X''$;
        $\pi := \pi \to [X' > X'']$;
  until X is a singlton;
end-procedure

3.3. TEST-L, TEST-H

The procedure TEST-L (TEST-H) takes as input a set X of coins and compares X with a heavy (light) unique set of coins of same size to determine whether X is heavy (light) uniform, or heavy (light) unique, or neither. According to the comparison, it will determine whether to continue the current cycle or not.

Procedure TEST-L(X): BOOLEAN;
choose Y from the identified coins, s.t. Y is heavily unique and $|Y| = |X|$;
Compare Y with X:
case $\|X\| > \|Y\|$:
$H := H \cup X$;
$U := U - X$;
if $|U| > 0$
then return TRUE;
else return FALSE;
case $\|X\| = \|Y\|$:
UNIQUE-L(X);
then return TRUE;
else return FALSE;
case $\|X\| < \|Y\|$:
return FALSE;
end-procedure

3.4. BACKTRACK-L, BACKTRACK-H

The procedure BACKTRACK-L (BACKTRACK-H) takes as input a set X of coins and the comparison path π for procedure DIG-L(X) (DIG-H(X)). It tries to explore as much information as possible from the comparison path π.

Procedure BACKTRACK-L(X, π): BOOLEAN
/* Consider the number of equality yields and inequality yields in π */
Case 1: π contains 0 equality yield.
/* This implies X is heavily unique */
Suppose π is ... $[\{y'\} > \{y\}]$.
$H := H \cup (X - \{y\})$;
$L := L \cup \{y\}$;
$U := U - X$;
if $|U| > 0$
then return TRUE;
else return FALSE;
Case 2: π contains 0 inequality yield:
/* This implies X is lightly uniform */
$L := L \cup X$;
$U := U - X$;
return FALSE;
Case 3: π contains 1 equality yield and 1^+ inequality yields.
/* consider the position of the equality in π */
Case 3.1: The equality is in the first position in π.
Suppose π is $[X'_1 = X_1] \to \cdots \to [\{y'\} > \{y\}]$.

/* this implies both X_1 and X'_1 are heavily unique. */
$H := H \cup (X_1 - \{y\})$;
$L := L \cup \{y\}$;
$U := U - X'_1$;
UNIQUE-L(X'_1);
if $|U| > 0$
then return TRUE;
else return FALSE;

Case 3.2: The equality yield is in some middle position in π.
Suppose π is $\cdots \to [X'_1 = X_1] \to \cdots \to [\{y'\} > \{y\}]$
/* This implies both X_1 and X'_1 are heavily unique. */
$H := H \cup (X_1 - \{y\})$;
$L := L \cup \{y\}$;
$U := U - X'_1$;
UNIQUE-L(X'_1);
return TEST-L($X - (X_1 \cup X'_1)$);

Case 3.3: The equality yield is in the last position in π.
if $|X| = 3$
then compare the other coin, say z, with an identified coin:
  if z is light
  then $L := L \cup X$;
    $U := U - X$;
  else $L := L \cup (X - \{z\})$;
    $H := H \cup \{z\}$;
    $U := U - X$;
  return FALSE;
if $|X| = 4$
then compare the other two coins, say z and z':
  if the yield is equality
  then $L := L \cup (X - \{z, z'\})$;
    $H := H \cup \{z, z'\}$;
  else *w.l.o.g., say* $\|z'\| > \|z\|$:
    $L := L \cup (X - \{z'\})$;
    $H := H \cup \{z'\}$;
  $U := U - X$;
  if $|U| > 0$
  then return TRUE;
  else return FALSE;
if $|X| > 4$
then suppose π is $\cdots \to [\{y'\} = \{y\}]$;
  $L := L \cup \{y, y'\}$;
  $U := U - \{y, y'\}$;
  Continue:= TEST-L($X - \{y, y'\}$);

if $|X|$ is not a power of 2 and Continue = FALSE
then return TEST-H($X - \{y, y'\}$);
else return Continue;

Case 4: π contains 2^+ equality yields and 1^+ inequality yields.

Case 4.1: π contains 2^+ consecutive equality yields in the end.
Suppose π is $\cdots \to [X'_1 > X_1] \to [\cdot = \cdot] \to \cdots \to [\cdot = \cdot]$, then
$L := L \cup X_1$
$U := U - X_1$;
return FALSE;

Case 4.2: π contains only 1 equality yield in the end
Suppose π is dots $\to [\{y'\} = \{y\}]$, then
$L := L \cup \{y, y'\}$;
$U := U - \{y, y'\}$;
if $|X|$ is a power of 2
then return TRUE;
else return TEST-H($X - \{y, y'\}$);

Case 4.3: π contains 2^+ consecutive inequality yields in the end
Suppose π is $\cdots \to [X'_1 = X_1] \to [\cdot > \cdot] \to \cdots \to [\{y'\} > \{y\}]$, then
$H := H \cup (X_1 - \{y\})$;
$L := L \cup \{y\}$;
$U := U - X_1$;
UNIQUE-L(X'_1);
return FALSE;

Case 4.4: π contains only 1 inequality yield in the end
Suppose π is $\cdots \to [X'_1 = X_1] \to [\{y'\} > \{y\}]$
if $|X'_1| = 1$
then X'_1 is light;
$L := L \cup X'_1 \cup \{y\}$;
$H := H \cup \{y'\}$;
$U := U - (X'_1 \cup X_1)$;
if $|X'_1| = 2$
then compare the two coins in X'_1, say z and z'.
w.l.o.g. suppose $\|z'\| > \|z\|$
$L := L \cup \{y, z\}$;
$H := H \cup \{y', z'\}$;
$U := U - (X_1 \cup X'_1)$;
if $|X|$ is a power of 2
then return FALSE;
else return TEST-H($X - (X_1 \cup X'_1)$);

end-procedure

3.5. Algorithm

The Algorithm A keeps doubly comparing until an inequality yield appears. Whenever an inequality yield appears, a pair of calls of DIG and BACKTRACK will be used to explore as much information as possible from the previous tests. In each while-loop, the following test will compare an unidentified set with a unique set. Notice that in the repeat-loop in Case 2, k is incremented only when Y is uniform. So in this repeat-loop, all other cases that return a TRUE value to *Continue* could be regarded as the '*interludes*' in the main algorithm.

Now we give the algorithm as follows.

Algorithm A.

```
input S;
L := ∅;
H := ∅;
U := S;
while |U| > 2 do
   X := one coin from U;
   U := U − X;
   k := 0;
   repeat
      Y := min {2^k, |S|} coins from U;
      X' := |Y| coins from X;
      compare Y with X';
      if ||X'|| = ||Y||
      then X := X ∪ Y;
         U := U − Y;
         k := k + 1;
   until ||X'|| = ||Y|| or U = ∅.
      Case 1: U = ∅:
         If there is any unidentified coin
         then use one more comparison to identify the type of X;
            if X is heavy
            then H := H ∪ X;
            else L := L ∪ X;
         else S is uniform.
      Case 2: ||X'|| > ||Y||:
         H := H ∪ X;
         if |Y| = 1
         then L := L ∪ Y;
            U := U − Y;
            if |U| = 0
            then stop;
```

```
        else break; /* restart another while-loop */
      else π := DIG-L(Y);
        Continue := BACKTRACK-L(Y, π);
        if Continue = TRUE
        then repeat
          Y := min{2^k, |U|} coins from U;
          Continue := TEST-L(Y);
          if Continue = TRUE
          then π := DIG-L(Y);
            Continue := BACKTRACK-L(Y, π);
          else if Y is uniform
          then k := k + 1;
        until Continue = FALSE;
    Case 3: ||X'|| < ||Y||.
      /* similar to case 2 */
  end-while
  if |U| = 1
  then compare U with an identified coin;
    if U is heavy
    then H := H ∪ U;
    else L := L ∪ U;
    U := ∅;
end-algorithm
```

4. Analysis

Next, we analyze Algorithm A.

For any set X coins, we denote:

$f(n, d) \triangleq \left(1 + \frac{1}{2}\log_2 2\right) d \log_2 \frac{n}{d} + 1.5d.$

$\bar{d} \triangleq \min\{d, n - d\}.$

$L_X \triangleq$ the light coins in X,

$H_X \triangleq$ the heavy coins in X,

$\bar{d}_X \triangleq$ the number of light coins in X.

Let S be the original input set of coins. From the symmetry of Algorithm A, we have

$$M_A(n:d) = M_A(n:n-d).$$

So we can denote

$$M_A(n:\bar{d}) \triangleq M_A(n:d) = M_A(n:n-d).$$

The following lemmas are fundamental to the inductive proof of our main result.

Lemma 4.1. *For any subset Y of X, the following hold:*

(1) $\bar{d}_Y + \bar{d}_{X-Y} \leqslant \bar{d}_X$;

(2) $f(|Y|, \bar{d}_Y) + f(|X - Y|, \bar{d}_{X-Y}) \leqslant f(|X|, \bar{d}_X)$.

Proof. (1) W.l.o.g., suppose $\bar{d}_X = |L_X|$. Since $\bar{d}_Y \leqslant |L_Y|$, $\bar{d}_{X-Y} \leqslant |L_{X-Y}|$ and $|L_Y| + |L_{X-Y}| = |L_X|$,

$$\bar{d}_Y + \bar{d}_{X-Y} \leqslant |L_Y| + |L_{X-Y}| = |L_X| = \bar{d}_X$$

(2) It follows immediately from (1) and Lemma 2.4. □

Lemma 4.2. *Consider a path π returned by DIG-L.*

(1) *Suppose $[X > Y]$ is in π, then $|H_X| \geqslant |H_Y|$;*

(2) *Suppose $[X = Y]$ is in π, then $|H_X| \geqslant |H_Y| - 1$, $|L_X| = |L_Y|$.*

Proof. (1): If $|X| = |Y|$, then $|H_X| > |H_Y|$.

If $|X| < |Y|$, then $|H_X| + 1 > |H_Y|$, hence $|H_X| \geqslant |H_Y|$.

If $|X| = |Y|$, then $|H_X| > |H_Y| + 1 > |H_Y|$.

(2): If $|X| = |Y|$, then $|H_X| > |H_Y|$ and $|L_X| = |L_Y|$.

If $|X| < |Y|$, then $|H_X| + 1 = |H_Y|$ hence $|H_X| \geqslant |H_Y| - 1$. It is obvious that $|L_X| = |L_Y|$.

If $|X| > |Y|$, then $|H_X| = |H_Y| + 1 \geqslant |H_Y| - 1$. It is obvious that $|L_X| = |L_Y|$. □

The following lemma considers the '*interlude*' cases at the end of which *Continue* is assigned with TRUE value.

Lemma 4.3. *Consider the procedure BACKTRACK-L(X, π). If Continue gets the TRUE return value, then the following hold:*

(1) *All coins in X are identified and there are still some unidentified coins in the input set of coins;*

(2) $|X| = 2^k$ *for some* $k > 1$;

(3) *X contains at most 3 light coins;*

(4) *The total number of tests to identify X, including the first test finding X containing light coins, the tests in DIG-L(X) and the tests in BACKTRACK-L(X, π), is not more than $f(|X|, \bar{d}_X)$;*

(5) *If X contains all the light coins in S, then the total number of tests to identify S is at most $f(|S|, |L_X|)$.*

Proof. We will consider all the possible cases in BACKTRACK-L(X, π) that could return a TRUE value to *Continue*. Since (1)–(3) are obvious in each case, here we only give the proof for (4) and (5). Let $|X| = 2^k$, $|S| = n$.

In Case 1:

(4): $\bar{d}_X = 1$. The total number of tests to identify X is $1 + k \leqslant f(2^k, 1)$ by Lemma 5.4
(1).

(5): It does not apply.

In Case 3.1:

(4): $\bar{d}_X = 2$. The total number of tests to identify X is $1 + k + \lceil \log_3 2^{k-1} \rceil \leqslant f(2^k, 2)$ by Lemma 5.5(1).

(5): $|L_X| = 2$. The total number of tests to identify S is

$$1 + k + \lceil \log_3 2^{k-1} \rceil + \lceil \log_2 (n - 2^k) \rceil \leqslant f(n, 2)$$

by Lemma 5.5(5).

In Case 3.2:

(4): In TEST-L($X - (X_1 \cup X'_1)$), if $X - (X_1 \cup X'_1)$, contains no light coin, then $\bar{d}_X = 2$ and the total number of tests to identify X is

$$1 + k + \lceil \log_3 |X'_1| \rceil + 1 \leqslant k + 2 + \lceil \log_3 2^{k-2} \rceil \leqslant f(2^k, 2)$$

by Lemma 5.5(2). If $X - (X_1 \cup X'_1)$ contains 1 light coin, then $\bar{d}_X = 3$. Suppose $|X_1| = |X'_1| = 2^{i-1}$ for some $i \leqslant k - 1$, then the total number of tests to identify X is

$$1 + k + \lceil \log_3 2^{i-1} \rceil + 1 + \lceil \log_3 (2^k - 2^i) \rceil \leqslant f(2^k, 3)$$

by Lemma 5.6(1).

(5): If X contains 2 light coins, then the total number of tests to identify S is

$$\begin{aligned} &1 + k + \lceil \log_3 |X'_1| \rceil + 1 + \lceil \log_2 (n - 2^k) \rceil \\ &\quad \leqslant k + 2 + \lceil \log_3 2^{k-2} \rceil + \lceil \log_2 (n - 2^k) \rceil \leqslant f(n, 2) \end{aligned}$$

by Lemma 5.5(6). If X contains 2 light coins, then the total number of tests to identify S is

$$1 + k + \lceil \log_3 2^{i-1} \rceil + 1 + \lceil \log_3 (2^k - 2^i) \rceil + \lceil \log_2 (n - 2^k) \rceil \leqslant f(2^k, 3)$$

by Lemma 5.6(3).

In Case 3.3:

(4): If $|X| = 4$, then the total number of tests to identify X is $1 + 2 + 1 = 4$. If X contains 2 light coins, then

$$f(4, 2) = (2 + \log_2 3) \log_2 \tfrac{4}{2} = 5 + \log_2 3 > 4.$$

If X contains 3 light coins, then

$$f(4, 1) = (1 + \tfrac{1}{2} \log_2 3) \log_2 4 + 1.5 = 3.5 \log_2 3 > 4.$$

If $|X| > 4$ and $X - \{y, y'\}$ contains no light coin, then $|L_X| = 2$ and the total number of tests to identify X is

$$1 + k + 1 = k + 2 \leqslant f(2^k, 2)$$

by Lemma 5.4(2).

If $|X| > 4$ and $X - \{y, y'\}$ contains 1 light coin, then $|L_X| = 3$ and the total number of tests to identify X is

$$1 + k + 1 + \lceil \log_3(2^k - 2) \rceil \leqslant f(2^k, 3)$$

by Lemma 5.6(1).

(5): If $|X| = 4$ and $|L_X| = 2$, then the total number of tests to identify S is

$$4 + \lceil \log_2(n - 4) \rceil \leqslant f(n, 2)$$

by Lemma 5.7(3).

If $|X| = 4$ and $|L_X| = 3$, then the total number of tests to identify S is

$$4 + \lceil \log_2(n - 4) \rceil \leqslant f(n, 2) \leqslant f(n, 3)$$

by Lemma 5.7(3).

If $|X| > 4$ and $|L_X| = 2$, then the total number of tests to identify S is

$$k + 2 + \lceil \log_2(n - 2^k) \rceil \leqslant f(n, 2)$$

by Lemma 5.5(6).

If $|X| > 4$ and $|L_X| = 3$, then the total number of tests to identify S is

$$k + 2 + \lceil \log_3(2^k - 2) \rceil + \lceil \log_2(n - 2^k) \rceil \leqslant f(n, 3)$$

by Lemma 5.6(4). □

The following lemma consider the '*restarting*' cases in the procedure BACKTRACK-L(X, π) at the end of which *Continue* is assigned with a FALSE value.

Lemma 4.4. *In BACKTRACK-L(X, π), a FALSE value is returned to Continue iff one of the following holds:*

(1) *X is identified and X is uniformly light with $|X| = 2^k$ for some k;*

(2) *X is identified and X contains at most 3 light coins, but there are no unidentified coins in the original input set;*

(3) *A subset $Y \subset X$ is identified, s.t. Y contains 4^+ light coins;*

(4) *A subset $Y \subset X$ is identified, s.t. Y contains 2 light coins and $\bar{d}_{X-Y} \geqslant 2$.*

Proof. We will consider all the possible cases in BACKTRACK-L(X, π) that could return a FALSE value to *Continue*.

In Case 1: This is (2).

In Case 2: This is (1).

In Case 3.1: This is (2).

In Case 3.2: If X contains 2 or 3 light coins, this is (2).

If X contains 4^+ light coins, then $X - (X_1 \cup X'_1)$ contains 2^+ light coins. We show this is (4) by showing that $|H_{X-(X_1 \cup X'_1)}| \geqslant 2$. Suppose π contains 3^+ inequality yields, then we could prove $|H_{X-(X_1 \cup X'_1)}| \geqslant 2$ by using Lemma 4.2 bottom-up. Now we assume π contains 2 inequality yields. Then π is of the form

$$[X_2' > X_2] \to [X_1' = X_1] \to [\{y'\} > \{y\}],$$

where $X = X_2' \cup X_2$ and $X_2 = X_1' \cup X_1$, thus $X_2' = X - (X_1' \cup X_1)$.

If $|X_1'| = |X_1| = 2$, then $|H_{X_1'}| = |H_{X_1}| = 1$ and $|H_{X_2}| = 2$. So $|H_{X_2'}| \geqslant ||H_{X_2}|| = 2$ by Lemma 4.2(1).

If $|X_1'| = 1$, then $|H_{X_1'}| = 0$ and $|H_{X_2}| = |H_{X_1}| = 1$.

If $|X_2'| \geqslant |X_2| = 3$, then $|H_{X_2'}| > |H_{X_2}| = 1$ and thus $|H_{X_2'}| \geqslant 2$.

If $|X_2'| < |X_2| = 3$, then $|X_2'| = 2$ and $|H_{X_2'}| = 1$. Thus $|L_{X_2'}| = 1$ and $|L_X| = |L_{X_2'}| + |L_{X_2}| = 1 + 2 = 3$, which contradicts to the condition that X contains 4^+ light coins.

In Case 3.3:

If $|X| \leqslant 4$, this is (2).

If $|X| > 4$ and X contains 2 or 3 light coins, this is (2).

If $|X| = 2^k$ for some $k > 2$ and X contains 4^+ light coins, then $|L_{X-\{y,y'\}}| \geqslant 2$. Also $|H_{X-\{y,y'\}}| \geqslant 2$ since π contains 2^+ inequality yields. So it is (4).

If $|X| > 4$ but $|X|$ is not a power of 2 and X contains 4^+ light coins and 0 or 1 heavy coins, then it is (2).

If $|X| > 4$ but $|X|$ is not a power of 2 and X contains 4^+ light coins and 2^+ heavy coins, then it is (4).

In Case 4.1: This is (3).

In Case 4.2:

If $|X|$ is a power of 2, then it is (2).

If $|X|$ is not a power of 2 and X contains at most 1 heavy coin, then it is (4).

If $|X|$ is not a power of 2 and X contains at most 2^+ heavy coins, then it is (3).

In Case 4.3: This is (4) which follows from Lemma 4.2(2).

In Case 4.4:

If $|X|$ is a power of 2, then it is (3).

If $|X|$ is not a power of 2 and X contains at most 2 heavy coins, then it is (4).

If $|X|$ is not a power of 2 and X contains at most 3^+ heavy coins, then it is (3). □

Lemma 4.5. $M_A(n:\bar{0}) = \lceil \log_2 n \rceil$.

Proof. Since the coin set is uniform, Algorithm A simply compares $\lceil \log_2 n \rceil$ times. □

Lemma 4.6. $M_A(n:\bar{1}) \leqslant 2\log_2 n$.

Proof. W.l.o.g., assume the unique coin is light. There are only 3 possible cases:

(1): The light coin is identified in the first test.

In this case the total number of tests is:

$$1 + 1 + \lceil \log_2(n-2) \rceil \leqslant 2\log_2 n$$

by Lemma 5.7(1).

(2): The light coin is identified in DIG-L on some 2^k coins for $1 \leqslant k \leqslant \lfloor \log_2 n \rfloor$. In this case the total number of tests is

$$1 + k + \lceil \log_2 (n - 2^k) \rceil \leqslant 2 \log_2 n$$

by Lemma 5.7(1).

(3): The light coin is identified in DIG-L on the last $n - 2^{\lfloor \log_2 n \rfloor}$ coins. In this case the total number of tests is

$$1 + \lfloor \log_2 n \rfloor + \lceil \log_2 (n - 2^{\lfloor \log_2 n \rfloor}) \rceil \leqslant 2 \log_2 n$$

by Lemma 5.7(1). □

Lemma 4.7. $M_A(n:\bar{2}) \leqslant 2 \log_2 n$.

Proof. W.l.o.g., assume there are exactly 2 light coins in the n ($n \geqslant 4$) coins.

If the first test identifies one light coin, then the total number of tests is at most

$$1 + M_A(n-2:\bar{1}) \leqslant 1 + 2\log_2(n-2) \leqslant f(n, 2)$$

by Lemmas 4.6 and 5.7(2).

Next we examine each case in the first call of BACKTRACK. If the first call of BACKTRACK is BACKTRACK-H, then it could happen only in case 2 in BACKTRACK-H. In the case, the total number of tests is at most

$$3 + 1 + \lceil \log_2 (n-4) \rceil = 4 + \lceil \log_2 (n-4) \rceil \leqslant f(n, 2)$$

by Lemma 5.7(3).

So now we suppose the first call of BACKTRACK is BACKTRACK-1. We examine each case in the following discussion.

In Case 1:

Only one light coin is identified. Then the other light coin will be identified by a call of UNIQUE-L(Y) on some set Y. Let $|Y| = m$, then the total number of tests is at most

$$1 + \lceil \log_3 m \rceil + M_A(n-m:\bar{1}) \leqslant 1 + \lceil \log_3 m \rceil + 2\log_2(n-m) \leqslant f(n, 2)$$

by Lemmas 4.6 and 5.7(2).

In Case 2:

If $n = 4$ then the total number of tests is 3 and

$$f(4, 2) = 5 + \log_2 3 \geqslant 3$$

If $n = 4$ then the total number of tests is

$$4 + \lceil \log_2 (n-4) \rceil \leqslant f(n, 2)$$

by Lemma 5.7(3).

In Case 3.1:

If a TRUE value is returned to *Continue*, then it follows from Lemma 4.3.

If a FALSE value is returned to *Continue*, then all coins are identified. Let $|X| = m$, then $n = 2^k + m$ for some k with $m \leqslant 2^k$. The total number of tests is

$$1 + k + \lceil \log_2 m \rceil + \lceil \log_3 \lfloor \tfrac{1}{2} m \rfloor \rceil \leqslant f(n, 2)$$

by Lemma 5.5(8).

In Case 3.2:

If a TRUE value is returned to *Continue*, then it follows from Lemma 4.3.

If a FALSE value is returned to *Continue*, then all coins are identified. Let $|X| = m$ and $|X_1| + |X'_1| = l$, then $n = 2^k + m$ for some k with $m \leqslant 2^k$. The total number of tests is

$$1 + k + \lceil \log_2 m \rceil + 1 + \lceil \log_3 \lfloor \tfrac{1}{2} m \rfloor \rceil \leqslant f(n, 2)$$

by Lemma 5.5(8).

In Case 3.3:

If a TRUE value is returned to *Continue*, then it follows from Lemma 4.3.

If a FALSE value is returned to *Continue*, then all coins are identified. Let $|X| = m$, then $n = 2^k + m$ for some k with $m \leqslant 2^k$. Consider the value of m as follows:

If $m = 3$, then $n = 2^k + 3$ and the total number of tests is

$$1 + k + 3 = k + 4 \leqslant f(2^k + 3, 2) = f(n, 2)$$

by Lemma 5.4(3).

If $m = 4$, then $n = 2^k + 4$ and the total number of tests is

$$1 + k + 3 = k + 4 \leqslant f(2^k + 4, 2) = f(n, 2)$$

by Lemma 5.4(3).

If $m > 4$, then the total number of tests is

$$1 + k + \lceil \log_2 m \rceil + 1 \leqslant f(n, 2)$$

by Lemma 5.5(8).

In Case 4:

This case cannot happen. □

Lemma 4.8. $M_A(n{:}\bar{d}) \leqslant f(n, \bar{d})$.

Proof. We prove by induction on $\bar{d}$. If $\bar{d} = 2$, this is exactly Lemma 4.7. Now we assume $\bar{d} \geqslant 3$.

If the first comparison is an inequality yield, let the two coins be x and x'. Then $\bar{d}_{S - \{x, x'\}} = \bar{d} - 1 \geqslant 2$. By induction, the total number of tests is at most

$$1 + M_A(n - 2, \bar{d} - 1) \leqslant f(2, 1) + f(n - 2, \bar{d} - 1) \leqslant f(n, \bar{d}).$$

Next we suppose the first comparison is equality yield. We consider the two cases in which the number of while-loops is one or greater than one.

(1) There is only one while-loop.

W.l.o.g., suppose the first inequality is ' > '. If there exists a call of UNIQUE-L(Y) with $|Y| \geqslant 2$ in the main algorithm, then $S - Y$ contains at least $|Y|$ heavy coins and at least $\bar{d} - 1$ light coins. Thus

$$\bar{d}_{S-Y} \geqslant \min\{|Y|, \bar{d} - 1\} \geqslant \min\{2, 3 - 1\} = 2.$$

Since Y is just an '*interlude*' and its existence does not affect the identification of other coins. So the total number of tests is at most

$$\begin{aligned} &1 + \lceil \log_3 |Y| \rceil + M_A(|S - Y| : \bar{d}_{S-Y}) \\ &\quad \leqslant f(|Y|, 1) + f(|S - Y|, \bar{d}_{S-Y}) \quad \text{(by induction and Lemma 5.4(1))} \\ &\quad = f(|Y|, \bar{d}_Y) + f(|S - Y|, \bar{d}_{S-Y}) \leqslant f(|S|, \bar{d}_S) \quad \text{(by Lemma 4.1).} \end{aligned}$$

We further suppose that there are no calls of UNIQUE-L in the main algorithm. If the first pair of calls of DIG-L(Y) and BACKTRACK(Y, π) in the main algorithm finds only one light coin, then $|Y| = 2^k$ for some $k > 0$ and only *case* 1 in BACKTRACK(Y, π) happens. Since there are no calls of UNIQUE-L, whether Y exists or does not affect all the following tests, that is, Y is also an '*interlude*'. By the same argument as above, $\bar{d}_{S-Y} \geqslant 2$ and the total number of tests is at most

$$\begin{aligned} &1 + \lceil \log_2 |Y| \rceil + M_A(|S - Y| : \bar{d}_{S-Y}) \\ &\quad \leqslant f(|Y|, 1) + f(|S - Y|, \bar{d}_{S-Y}) \quad \text{(by induction and Lemma 5.4(1))} \\ &\quad = f(|Y|, \bar{d}_Y) + f(|S - Y|, \bar{d}_{S-Y}) \leqslant f(|S|, \bar{d}_s) \quad \text{(by Lemma 4.1).} \end{aligned}$$

Next we can assume there are no calls of UNIQUE-L and the first pair of calls of DIG-L and BACKTRACK-L finds 2^+ light coins. Suppose there are more than one such pair, then each pair of DIG-L and BACKTRACK-L will find 2^+ light coins and can be regarded as independent. Let the first pair of calls be DIG-L(Y) and BACKTRACK(Y, π). Then $d_{S-Y} \geqslant 2$ since $S - Y$ contains at least $|Y| \geqslant 2$ heavy coins and 2^+ light coins. By Lemma 4.3, the total number of tests to identify Y is at most $f(|Y|, \bar{d}_Y)$. By induction, the number of tests to identify $S - Y$ is at most $f(|S - Y|, \bar{d}_{S-Y})$. So the total number of tests to identify S is at most $f(|S|, \bar{d}_S)$ by Lemma 4.1.

Next we suppose there are no calls of UNIQUE-L and there is only one pair of calls of DIG-L and BACKTRACK. If this pair is in the middle of the main algorithm, then by Lemma 4.3, the total number of tests is also at most $f(|S|, \bar{d}_S)$. So we only need to consider the case in which the pair of calls happen last in the main algorithm. We examine all the possible cases in BACKTRACK(X, π).

In Case 2:

Suppose $|X| = 2^l$ and $|S| = 2^k + 2^l$ with $2 \leqslant l \leqslant k$. Then $\bar{d}_S = 2^l$ and the total number of tests to indentify S is at most

$$\begin{aligned} 1 + k + l \leqslant 2k + 1 &\leqslant f(2^k + 2, 2) \quad \text{(by Lemma 5.4(3))} \\ &\leqslant f(2^k + 2^l, 2) \quad \text{(by Lemma 5.1)} \end{aligned}$$

$$\leqslant f(2^k + 2^l, 2^l) \quad \text{(by Lemma 5.1)}$$

$$= f(|S|, \bar{d}_S)$$

In Case 3.2:

X must contains 3 light coins and thus $\bar{d}_S = 3$. Let $|X| = m$, then $|S| = n = 2^k + m$ for some k with $2^k \geqslant m$. Suppose $|X'_1 \cup X_1| = l$, then the total number of tests to identify S is at most

$$1 + k + \lceil \log_3 m \rceil + \lceil \log_3 \lfloor \tfrac{1}{2} l \rfloor \rceil + \lceil \log_3 (m - l) \rceil \leqslant f(n, 3)$$

by Lemma 5.6(4).

In Case 3.3:

X must contain all $\bar{d}_S \geqslant 3$ light coins. Let $|X| = m$, then $|S| = n = 2^k + m$ for some k with $2^k \geqslant m$.

If $m = 3$, then $n = 2^k + 3$ and the total number of tests is

$$1 + k + 3 = k + 4 \leqslant f(2^k + 3, 2) = f(n, 2) \leqslant f(n, 3)$$

by Lemma 5.4(3).

If $m = 4$, then $n = 2^k + 4$ and the total number of tests is

$$1 + k + 3 = k + 4 \leqslant f(2^k + 4, 2) = f(n, 2) \leqslant f(n, 3)$$

by Lemma 5.4(3).

If $m > 4$ and X contains 3 light coins, then the total number of tests to identify S is at most

$$1 + k + \lceil \log_2 m \rceil + 1 + \lceil \log_3 (m - 2) \rceil \leqslant f(n, 3)$$

by Lemma 5.6(5).

If $m > 4$ and X contains no heavy coins, then the total number of tests to identify S is at most

$$1 + k + \lceil \log_2 m \rceil + 1 + 1 = k + 3 + \lceil \log_2 m \rceil \leqslant f(2^k + m, m)$$

by Lemma 5.6(6).

If $m > 4$ and X contains only one heavy coin, then the total number of tests to identify S is at most

$$1 + k + \lceil \log_2 m \rceil + 1 + \lceil \log_3 (m - 2) \rceil$$

$$= k + 2 + \lceil \log_2 m \rceil + \lceil \log_3 (m - 2) \rceil \leqslant f(2^k + m, m - 1)$$

by Lemma 5.6(7).

In Case 4.2:

$|X|$ cannot be a power of 2. Let $|X| = m$, then $m \leqslant 5$ and $|S| = n = 2^k + m$ for some k with $2^k \geqslant m$.

If X contains no heavy coins, then $\bar{d}_S = m$. The total number of tests to identify S is at most

$$\begin{aligned}1 + k + \lceil \log_2 m \rceil + 1 \leqslant 2k + 2 &\leqslant f(2^k + 2, 2) \quad \text{(by Lemma 5.4(3))}\\ &\leqslant f(2^k + m, 2) \quad \text{(by Lemma 5.1)}\\ &\leqslant f(2^k + m, m) \quad \text{(by Lemma 5.1).}\end{aligned}$$

If X contains 1 heavy coin, then $\bar{d}_S = m - 1 \geqslant 4$. The total number of tests to identify S is at most

$$\begin{aligned}&1 + k + \lceil \log_2 m \rceil + 1 + \lceil \log_3 (m-2) \rceil \leqslant 2k + 2 + \lceil \log_3 (m-2) \rceil\\ &\quad\leqslant f(2^k + 2, 2) + f(m-2, 1) \quad \text{(by Lemma 5.4(1) and (3))}\\ &\quad\leqslant f(2^k + m, 3) \quad \text{(by Lemma 2.3)}\\ &\quad\leqslant f(2^k + m, m) \quad \text{(by Lemma 5.1).}\end{aligned}$$

In Case 4.4:

$|X|$ cannot be a power of 2. Let $|X| = m$, then $m \leqslant 5$ and $|S| = n = 2^k + m$ for some k with $2^k \geqslant m$.

If $|X'_1| = 1$ and X contains 1 heavy coin then $\bar{d}_S = m - 1$ and the total number of tests to identify S is at most

$$1 + k + \lceil \log_2 m \rceil + 1 \leqslant 2k + 2 \leqslant f(2^k + m, m-1)$$

by the same argument as in Case 4.2.

If $|X'_1| = 1$ and X contains 2 heavy coins then $\bar{d}_S = m - 2$ and the total number of tests to identify S is at most

$$\begin{aligned}&1 + k + \lceil \log_2 m \rceil + 1 + \lceil \log_3 (m-3) \rceil \leqslant 2k + 2 + \lceil \log_3 (m-3) \rceil\\ &\quad\leqslant f(2^k + 2, 2) + f(m-3, 1)\\ &\quad\leqslant f(2^k + m - 1, 3) \leqslant f(2^k + m, m-2)\end{aligned}$$

by the same argument as in Case 4.2.

If $|X'_1| = 1$ and X contains 2 heavy coins then $\bar{d}_S = m - 2$ and the total number of tests to identify S is at most

$$\begin{aligned}&1 + k + \lceil \log_2 m \rceil + 1 + \lceil \log_3 (m-3) \rceil\\ &\quad\leqslant 2k + 2 + \lceil \log_3 (m-3) \rceil\\ &\quad\leqslant f(2^k + 2, 2) + f(m-3, 1) \leqslant f(2^k + m - 1, 3) \leqslant f(2^k + m, m-2)\end{aligned}$$

by the same argument as in Case 4.2.

If $|X'_1| = 2$ and X contains 1 heavy coin then $\bar{d}_S = m - 1$ and the total number of tests to identify S is at most

$$\begin{aligned}&1 + k + \lceil \log_2 m \rceil + 1 + 1 \leqslant 2k + 3 \leqslant f(2^k + 2, 2) \leqslant f(2^k + m, 2)\\ &\quad\leqslant f(2^k + m, m-1)\end{aligned}$$

by the same argument as in Case 4.2.

If $|X'_1| = 2$ and X contains 2 heavy coins then $\bar{d}_S = m - 2$ and the total number of tests to identify S is at most

$$1 + k + \lceil \log_2 m \rceil + 1 + 1 + \lceil \log_3 (m-4) \rceil \leqslant 2k + 3 + \lceil \log_3 (m-4) \rceil$$

$$\leqslant f(2^k + 2, 2) + f(m-4, 1) \leqslant f(2^k + m - 2, 3) \leqslant f(2^k + m, m - 2)$$

by the same argument as in Case 4.2.

(2) The algorithm contains more than one while-loop.

Consider the first while-loop. W.l.o.g., assume the first inequality comparison yield is '$>$'. By the same argument as in (1), we can assume that there are no calls of UNIQUE-L in the main algorithm. Also by Lemmas 4.1 and 4.3, we can assume that there are no '*interlude*' cases. We consider each case in BACKTRACK-L(X, π) that could return a FALSE value to *Continue* and yet at the end of which there are still some unidentified coins. Let Z be the set of coins identified in the first while-loop and suppose that before testing X we already have identified 2^k heavy coins for some $k \geqslant 1$.

In Case 2:

In this case, X is lightly uniform and $|X| = 2^k$.

If $\bar{d}_{S-Z} \geqslant 2$, then by induction the total number of tests to identify $S - Z$ is at most $f(|S - Z|, \bar{d}_{S-Z})$. The total number of tests to identify Z is at most

$$2k + 1 \leqslant f(2^k + 2, 2) \leqslant f(2^{k+1}, 2) \leqslant f(2^{k+1}, 2^k) = f(|Z|, \bar{d}_Z).$$

So by Lemma 4.1, the total number of tests to identify S is at most $f(|S|, \bar{d}_S)$.

If $\bar{d}_{S-Z} = 1$ and $k \leqslant 2$, then the total number of tests to identify S is at most

$$2k + 1 + M_A(|S - Z| : \bar{1}) \leqslant 2k + 1 + 2\log(n - 2^{k+1})$$

$$\leqslant f(2^k + 2 + n - 2^{k+1}, 4) \quad \text{(by Lemma 5.4(5))}$$

$$\leqslant f(n, 4) \leqslant f(n, 2^k + 1) = f(|S|, \bar{d}_S).$$

If $\bar{d}_{S-Z} = 1$ and $k = 1$, then the total number of tests to identify S is at most

$$3 + M_A(n - 4 : \bar{1}) \leqslant 3 + 2\log(n-4) \leqslant f(n, 2) \quad \text{(by Lemma 5.7(2))}$$

$$\leqslant f(n, 2) \leqslant f(n, 3) = f(|S|, \bar{d}_S).$$

If $\bar{d}_{S-Z} = 0$, then $k \leqslant 2$ and the total number of tests to identify S is at most

$$2k + 1 + \lceil \log_2 (n - 2^{k+1}) \rceil + 1 \leqslant f(2^k + 2 + n - 2^{k+1}, 3) \quad \text{(by Lemma 5.4(4))}$$

$$\leqslant f(n, 3) \leqslant f(n, 2^k) = f(|S|, \bar{d}_S).$$

In Case 3.2:

In this case X must contain 4^+ light coins and by Lemma 4(4) $\bar{d}_{X-(X_1 \cup X'_1)} \geqslant 2$, hence $\bar{d}_{S-Z} \geqslant 2$. Suppose $|X| = m$ and $|X_1 \cup X'_1| = l$, then the total number of tests to identify Z is at most

$$1 + k + \lceil \log_2 m \rceil + \lceil \log_3 \lfloor \tfrac{1}{2} l \rfloor \rceil + 1 \leqslant f(2^k + l, 2) \quad \text{(by Lemma 5.5(8))}$$

$$= f(|Z|, \bar{d}_Z).$$

So by Lemma 4.1, the total number of tests to identify S is at most $f(|S|, \bar{d}_S)$.

In Case 3.3:

In this case X must contain 4^+ light coins and 2^+ heavy coins. By Lemma 4(4) $\bar{d}_{X-\{x,y\}} \geqslant 2$, hence $\bar{d}_{S-Z} \geqslant 2$. Suppose $|X| = m$, then the total number of tests to identify Z is at most

$$1 + k + \lceil \log_2 m \rceil + 2 \leqslant 2k + 3 \leqslant f(2^k + 2, 2) \quad \text{(by Lemma 5.4(3))}$$

$$= f(|Z|, \bar{d}_Z).$$

So by Lemma 4.1, the total number of tests to identify S is at most $f(|S|, \bar{d}_S)$.

In Case 4.1:

In this case, $|X_1 \cup X'_1| = 2^l$ for some $l \geqslant 2$. The proof is similar to the proof in Case 2.

In Case 4.2:

In this case X must contain 4^+ light coins. The proof is similar to the proof in Case 3.2.

In Case 4.3:

In this case X must contain 3^+ heavy coins by Lemma 4(4) $\bar{d}_{X-(X_1 \cup X'_1)} \geqslant 2$, hence $\bar{d}_{S-Z} \geqslant 2$. Suppose $|X| = m$.

If $|X'_1| = 1$, then the total number of tests to identify Z is at most

$$1 + k + \lceil \log_2 m \rceil + 1 \leqslant 2k + 2 \leqslant f(2^k + 2, 2) \quad \text{(by Lemma 5.4(3))}$$

$$\leqslant f(2^k + 3, 2) = f(|Z|, \bar{d}_Z).$$

So by Lemma 4.1, the total number of tests to identify S is at most $f(|S|, \bar{d}_S)$.

If $|X'_1| = 2$, then the total number of tests to identify Z is at most

$$1 + k + \lceil \log_2 m \rceil + 2 \leqslant 2k + 3 \leqslant f(2^k + 2, 2) \quad \text{(by Lemma 5.4(3))}$$

$$\leqslant f(2^k + 4, 2) = f(|Z|, \bar{d}_Z).$$

So by Lemma 4.1, the total number of tests to identify S is at most $f(|S|, \bar{d}_S)$.

Thus in either case, the total number of tests to identify S is at most $f(|S|, \bar{d}_S)$. □

Now we can prove our main result:

Theorem 4.9. *For* $0 < d < n$,

$$M_A(n : \bar{d}) \leqslant (\log_2 3 + \tfrac{1}{2}) M(n : d) + 4.$$

Proof. By Lemma 2.3, $M(n:1) = \lceil \log_2 n \rceil$.

By Lemma 4.6, $M_A(n:\bar{1}) \leqslant 2\log_2 n \leqslant (\log_2 3 + \frac{1}{2})\log_2 n$. So

$$M_A(n:\bar{1}) \leqslant (\log_2 3 + \tfrac{1}{2}) M(n:1).$$

For $\bar{d} \geqslant 2$, by Lemm 2.1 we have

$$(\log_2 3 + \tfrac{1}{2}) M(n:d) \geqslant (\log_2 3 + \tfrac{1}{2}) M(n,d) \geqslant f(n,d) + h(d),$$

where

$$f(n,d) = (1 + \tfrac{1}{2}\log_3 2) d \log_2 \tfrac{n}{d} + 1.5d,$$

$$h(d) = (1 + \tfrac{1}{2}\log_3 2)\left[\left(\log_2 \frac{e\sqrt{3}}{2} - \frac{1.5}{1 + \frac{1}{2}\log_3 2}\right) d - \frac{1}{2}\log_2 d - 0.567 - \frac{1}{2}\log_2 3\right].$$

Since $h'(d) = 0$ iff $d = 7.6\ \ldots$,

$$h(d) \geqslant \min\{h(7), h(8)\} = -3.8 \cdots \geqslant -4.$$

Thus by Lemmas 4.8 and 5.2,

$$(\log_2 3 + \tfrac{1}{2}) M(n:d) \geqslant f(n,d) - 4 \geqslant f(n:\bar{d}) - 4 \geqslant M_A(n:\bar{d}) - 4.$$

So $M_A(n:\bar{d}) \leqslant (\log_2 3 + \frac{1}{2}) M(n:d) + 4$. □

Appendix

In this section, we present some properties and inequalities used by our algorithm analysis.

Let $f(n,d) = (1 + \dfrac{1}{2}\log_3 2) d \log_2 \dfrac{n}{d} + 1.5d$.

Lemma 1. *f is an increasing function of n and d for $d \leqslant \frac{n}{2}$.*

Proof. It is obvious that f is an increasing function of n.

Since for $d \leqslant \frac{1}{2}n$,

$$\begin{aligned} f'_d(n,d) &= (1 + \frac{1}{2}\log_3 2)\log_2 \frac{n}{d} - (1 + \frac{1}{2}\log_3 2)\log_2 e + 1.5 \\ &= \left(1 + \frac{1}{2}\log_3 2\right)\log_2 \frac{n}{d \cdot e} + 1.5 \\ &\geqslant \left(1 + \frac{1}{2}\log_3 2\right)\log_2 \frac{2}{e} + 1.5 \geqslant 0. \end{aligned}$$

f is also an increasing function of d for $d \leqslant \frac{1}{2}n$. □

Lemma 2. *For $d \leqslant \frac{1}{2}n$,*

$$f(n,d) \leqslant f(n,n-d).$$

Proof. *Let $g(d) = f(n,d) - f(n,n-d)$, then*

$$\begin{aligned} g'(d) &= \left(1 + \frac{1}{2}\log_3 2\right)\log_2 \frac{n}{d\cdot e} + 1.5 + \left(1 + \frac{1}{2}\log_3 2\right)\log_2 \frac{n}{(n-d)\cdot e} + 1.5 \\ &= \left(1 + \frac{1}{2}\log_3 2\right)\log_2 \frac{n^2}{d(n-d)\cdot e^2} + 3 \\ &= \left(1 + \frac{1}{2}\log_3 2\right)\log_2 \frac{n^2}{[(d+n-d)/2]^2\cdot e^2} + 3 \\ &= 2\left[\left(1 + \frac{1}{2}\log_3 2\right)\log_2 \frac{2}{e} + 1.5\right] > 0. \end{aligned}$$

So g is strictly increasing. Since $g(\frac{1}{2}n) = 0$, for $d \leqslant \frac{1}{2}n$, $g(d) \leqslant g(\frac{1}{2}n) = 0$. This means for $d \leqslant \frac{1}{2}n$, $f(n,d) \leqslant f(n,n-d)$. □

Lemma 3. *Let $g(k,m) = f(2^k + m, d)$ for any fixed $d \geqslant 2$.*

(1) *If $m \geqslant 2^k$, then $g'_m(k,m) > 1/(m\ln 2)$. This implies that g increases faster than $\log_2 m$ as m increases.*

(2) *If $m \leqslant 2^k$, then $g'_k(k,m) > 1$. This implies that g increases faster than k as k increases.*

Proof. (1) For $d \geqslant 2$ and $m \geqslant 2^k$,

$$\begin{aligned} g'_m(k,m) &= \left(1 + \frac{1}{2}\log_3 2\right)d\cdot\frac{1}{2^k+m}\cdot\frac{1}{\ln 2} \geqslant \left(1 + \frac{1}{2}\log_3 2\right)2\cdot\frac{1}{m+m}\cdot\frac{1}{\ln 2} \\ &> \frac{1}{m\ln 2}. \end{aligned}$$

(2) For $d \geqslant 2$ and $m \leqslant 2^k$,

$$g'_k(k,m) = \left(1 + \frac{1}{2}\log_3 2\right)d\cdot\frac{2^k}{2^k+m} \geqslant \left(1 + \frac{1}{2}\log_3 2\right)2\cdot\frac{2^k}{2^k+2^k} > 1. \quad \square$$

Lemma 4.

(1) $\lceil \log_2 m \rceil + 1 \leqslant f(m,1)$.

(2) $2\log_2 m \leqslant f(m,2)$.

(3) $2k + 3 \leqslant f(2^k + 2, 2)$.

(4) $2k + \lceil \log_2 m \rceil + 2 \leqslant f(2^k + 2 + m, 3)$.

(5) $2k + 2\log_2 m + 1 \leqslant f(2^k + 2 + m, 4)$.

(6) $2k + 2 + \lceil \log_3 \lfloor \frac{1}{2}l \rfloor \rceil \leqslant f(2^k + l, 2)$.

Proof. (1): If $m = 1$, then

$$f(m,1) - (\lceil \log_2 m \rceil + 1) = 1.5 - 1 > 0.$$

If $m = 2$, then

$$f(m,1) - (\lceil \log_2 m \rceil + 1) = \tfrac{1}{2}\log_3 2 + 2.5 - 2 > 0.$$

For $m \geqslant 3$,

$$f(m,1) - (\lceil \log_2 m \rceil + 1) \geqslant f(m,1) - \log_2 m - 2$$

$$= (1 + \tfrac{1}{2}\log_3 2)\log_2 m - \log_2 m - 0.5 = \tfrac{1}{2}\log_3 m - 1 \geqslant 0.$$

Thus $\lceil \log_2 m \rceil + 1 \leqslant f(m,1)$.

(2): It follows that

$$f(m,2) - 2\log_2 m = (2 + \log_3 2)\log_2 \tfrac{1}{2}m + 3 - 2\log_2 m$$

$$= \log_3 m - \log_3 2 + 1 > 0.$$

(3): Let $g(k) = f(2^k + 2, 2) - (2k + 3)$. Then

$$g(k) = (2 + \log_3 2)\log_2 \tfrac{1}{2}(2^k + 2) - 2k = (2 + \log_3 2)\log_2 2^{k-1} + 1 - 2k$$

and

$$g'(k) = (2 + \log_3 2)\frac{2^{k-1}}{2^{k-1} + 1} - 2.$$

So

$$g'(k) \geqslant 0 \iff 2^{k-2} \geqslant \log_2 3 \iff k \geqslant 3.$$

Since

$$g(3) = (2 + \log_3 2)\log_2 5 - 6 \approx 0.10 > 0$$

$$g(2) = (2 + \log_3 2)\log_2 3 - 4 = 2\log_2 3 - 3 = \log_2 9 - \log_2 8 > 0$$

$$g(1) = 2 + \log_3 2 - 2 = \log_3 2 > 0.$$

We have $g(k) > 0$ for all $k > 0$. Thus $2k + 3 \leqslant f(2^k + 2, 2)$.

(4) and (5) follow immediately from (1)–(3).

(6): Let $g(l) = f(2^k + l, 2) - (2k + 3 + \log_3 \frac{l}{2})$, then

$$g(l) = (2 + \log_3 2)\log_2 \tfrac{1}{2}(2^k + l) - 2k - \log_3 l + \log_3 2$$

$$= (2 + \log_3 2)\log_2 2^k + l - 2k - \log_3 l - 2$$

and

$$g'(l) = \frac{2 + \log_3 2}{(2^k + l)\ln 2} - \frac{1}{l \ln 3}.$$

So

$$g'(l) \geqslant 0 \Leftrightarrow (2+\log_3 2)\log_2 3 \cdot l \geqslant 2^k + l \Leftrightarrow (2\log_2 3 + 1)\cdot l \geqslant 2^k + l$$
$$\Leftrightarrow l \geqslant 2^{k-1}\log_3 2.$$

Since

$$\begin{aligned} g(2^{k-1}\log_3 2) &= (2+\log_3 2)\log_2(2^k + 2^{k-1}\log_3 2) - 2k - \log_3(2^{k-1}\log_3 2) - 2 \\ &= (2+\log_3 2)[k + \log_2(1+\tfrac{1}{2}\log_3 2)] - 2k - (k-1)\log_3 2 \\ &\quad - \log_3\log_3 2 - 2 \\ &= (2+\log_3 2)\log_2(1+\tfrac{1}{2}\log_3 2) + \log_3 2 - \log_3\log_3 2 - 2 \\ &\approx 0.09 > 0, \end{aligned}$$

for all l,

$$g(l) \geqslant g(k, 2^{k-1}\log_3 2) > 0.$$

This implies

$$2k + 2 + \lceil \log_3 \lfloor \tfrac{1}{2} l \rfloor \rceil \leqslant f(2^k + l, 2). \qquad \square$$

Lemma 5. (1) $K + 1 + \lceil \log_3 2^{k-1} \rceil \leqslant f(2^k, 2)$, *for all* $k \geqslant 1$.

(2) $K + 2 + \lceil \log_3 2^{k-2} \rceil \leqslant f(2^k, 2)$, *for all* $k \geqslant 2$.

(3) $2K + 2 + \lceil \log_3 2^{k-1} \rceil \leqslant f(2^{k+1}, 2)$, *for all* $k \geqslant 1$.

(4) $2K + 3 + \lceil \log_3 2^{k-2} \rceil \leqslant f(2^{k+1}, 2)$, *for all* $k \geqslant 2$.

(5) $K + 1 + \lceil \log_3 2^{k-1} \rceil + \lceil \log_2 m \rceil \leqslant f(2^k + m, 2)$, *for all* $k \geqslant 1$ *and* $m \geqslant 2^k$.

(6) $K + 2 + \lceil \log_3 2^{k-2} \rceil + \lceil \log_2 m \rceil \leqslant f(2^k + m, 2)$, *for all* $k \geqslant 2$ *and* $m \geqslant 2^k$.

(7) $K + 1 + \lceil \log_2 m \rceil + \lceil \log_3 \lfloor \tfrac{1}{2} l \rfloor \rceil \leqslant f(2^k + l, 2)$, *for all* $m \leqslant 2^k$ *and* $l \leqslant \lceil \tfrac{1}{2} m \rceil$.

(8) $K + 2 + \lceil \log_2 m \rceil + \lceil \log_3 \lfloor \tfrac{1}{2} l \rfloor \rceil \leqslant f(2^k + m, 2)$, *for all* $m \leqslant 2^k$ *and* $l \leqslant \lceil \tfrac{1}{2} m \rceil$.

Proof. (1): For $k \geqslant 1$,

$$\begin{aligned} f(2^k, 2) - (K + 1 + \lceil \log_3 2^{k-1} \rceil) &\geqslant (2+\log_3 2)(k-1) + 3 \\ &\quad - (k + 2 + k\log_3 2 - \log_3 2) = k - 1 \geqslant 0. \end{aligned}$$

Hence for $k \geqslant 1$,

$$k + 1 + \lceil \log_3 2^{k-1} \rceil \leqslant f(2^k, 2).$$

(2): For $k \geqslant 2$,

$$\begin{aligned} f(2^k, 2) - (K + 2 + \lceil \log_3 2^{k-2} \rceil) &\geqslant (2+\log_3 2)(k-1) + 3 \\ &\quad - (k + 3 + k\log_3 2 - 2\log_3 2) \\ &= k - 2 + \log_3 2 > 0. \end{aligned}$$

Hence for $k \geqslant 2$,

$$k + 2 + \lceil \log_3 2^{k-2} \rceil \leqslant f(2^k, 2).$$

(3): For $k \geqslant 1$,

$$f(2^{k+1}, 2) - (2k + 2 + \lceil \log_3 2^{k-1} \rceil) \geqslant (2 + \log_3 2)k + 3$$
$$- (2k + 3 + k\log_3 2 - \log_3 2) = \log_3 2 \geqslant 0.$$

Hence for $k \geqslant 1$,

$$2k + 2 + \lceil \log_3 2^{k-1} \rceil \leqslant f(2^{k+1}, 2).$$

(4): For $k \geqslant 2$,

$$f(2^{k+1}, 2) - (2K + 3 + \lceil \log_3 2^{k-2} \rceil) \geqslant (2 + \log_3 2)k + 3$$
$$- (2k + 4 + k\log_3 2 - 2\log_3 2)$$
$$= 2\log_3 2 - 1 > 0.$$

Hence for $k \geqslant 2$,

$$2k + 3 + \lceil \log_3 2^{k-2} \rceil \leqslant f(2^{k+1}, 2).$$

(5) and (6): By Lemma 5.3(1), for $m \geqslant 2^k, f(2^k + m, 2)$ increases faster than $\log_2 m$ as m increases. So (5) and (6) hold for $m \geqslant 2^k$ iff they hold for $m = 2^k$, which are (3) and (4), respectively.

(7): By Lemma 5.3(2), for $k > \log_2 m$, $f(2^k + m, 2)$ increases faster than k as k increases. So we only need to prove that (7) holds for $k = \lceil \log_2 m \rceil$.

$$f(2^k + l, 2) - (K + 1 + \lceil \log_2 m \rceil + \lceil \log_3 \lfloor \tfrac{1}{2} l \rfloor \rceil)$$
$$\geqslant (2 + \log_3 2)\log_2 \tfrac{1}{2}(2^k + l) + 3 - (k + 1 + k + 1 + \log_3 \tfrac{1}{2} l)$$
$$= (2 + \log_3 2)\left[k + \log_2\left(\frac{1}{2} + \frac{l}{2^{k+1}}\right)\right] - \left(2k + k\log_3 2 + \log_3 \frac{l}{2^{k+1}}\right) + 1$$
$$= (2 + \log_3 2)\log_2\left(\frac{1}{2} + \frac{l}{2^{k+1}}\right) - \log_3 \frac{1}{2^{k+1}} + 1 > 0.$$

(8): By Lemma 5.3(2), for $k > \log_2 m$, $f(2^k + m, 2)$ increases faster than k as k increases. So we only need to prove (8) holds for $k = \lceil \log_2 m \rceil$.

$$f(2^k + l, 2) - (K + 2 + \lceil \log_2 m \rceil + \lceil \log_3 \lfloor \tfrac{1}{2} l \rfloor \rceil)$$
$$\geqslant (2 + \log_3 2)\log_2 \tfrac{1}{2}(2^k + m) + 3 - (k + 2 + k + 1 + \log_3 \tfrac{1}{2} l)$$
$$= (2 + \log_3 2)\left[k + \log_2\left(\frac{1}{2} + \frac{m}{2^{k+1}}\right)\right] - \left(2k + k\log_3 2 + \log_3 \frac{l}{2^{k+1}}\right)$$
$$= (2 + \log_3 2)\log_2\left(\frac{1}{2} + \frac{m}{2^{k+1}}\right) - \log_3 \frac{l}{2^{k+1}} > 0. \qquad \square$$

Lemma 6. (1) $K + 2 + \lceil \log_3 2^{i-1} \rceil + \lceil \log_3 (2^k - 2^i) \rceil \leqslant f(2^k, 3)$, *for all* $k \geqslant 3$ *and* $i \leqslant k-1$.

(2) $2K + 3 + \lceil \log_3 2^{i-1} \rceil + \lceil \log_3 (2^k - 2^i) \rceil \leqslant f(2^{k+1}, 3)$, *for all* $k \geqslant 2$ *and* $i \leqslant k-1$.

(3) $K + 2 + \lceil \log_2 m \rceil + \lceil \log_3 2^{i-1} \rceil + \lceil \log_3 (2^k - 2^i) \rceil \leqslant f(2^k + m, 3)$, *for all* $k \geqslant 3$, $i \leqslant k-1$ *and* $m \geqslant 2^k$.

(4) $K + 2 + \lceil \log_2 m \rceil + \lceil \log_3 \lfloor \frac{1}{2} l \rfloor \rceil + \lceil \log_3 (m - l) \rceil \leqslant f(2^k + l, 3)$, *for all* $k \geqslant 3$, $m \geqslant 2^k$ *and* $l \leqslant \lceil \frac{1}{2} m \rceil$.

(5) $k + 2 + \lceil \log_2 m \rceil + \lceil \log_3 (m-2) \rceil \leqslant f(2^k + m, 3)$, *for* $4 < m \leqslant 2^k$.

(6) $k + 3 + \lceil \log_2 m \rceil \leqslant f(2^k + m, m)$, *for* $4 < m \leqslant 2^k$.

(7) $k + 3 + \lceil \log_2 m \rceil + \lceil \log_3 (m-2) \rceil \leqslant f(2^k + m, m-1)$, *for* $4 < m \leqslant 2^k$.

Proof. (1): For $k \geqslant 3$ and $i \leqslant k-1$,

$$\begin{aligned}
&f(2^k, 3) - (K + 2 + \lceil \log_3 2^{i-1} \rceil + \lceil \log_3 (2^k - 2^i) \rceil) \\
&\quad \geqslant (1 + \tfrac{1}{2}\log_3 2) 3 \log_2 \frac{2^k}{3} + 4.5 - (k + 4 + \log_3 2^i + \log_3 (2^k - 2^i) - \log_3 2) \\
&\quad = (3 + \tfrac{3}{2}\log_3 2) k - (3 \log_2 3 + 1.5) + 4.5 - \{k + 4 + \log_3 [2^i \cdot (2^k - 2^i)] \\
&\qquad + 4 - \log_3 2\} \\
&\quad \geqslant (3 + \tfrac{3}{2}\log_3 2) k - [k + \log_3 (2^{k-1} \cdot 2^{k-1}) + 4 - \log_3 2] \\
&\qquad - (3 \log_2 3 + 1 - \log_3 2) \\
&\quad = (3 + \tfrac{3}{2}\log_3 2) k - (k + 2 \log_3 2 \cdot k - 2 \log_3 2) - (3 \log_2 3 + 1 - \log_3 2) \\
&\quad = (2 - \tfrac{1}{2}\log_3 2) k - (3 \log_2 3 + 1 - 3 \log_3 2) \\
&\quad \geqslant (2 - \tfrac{1}{2}\log_3 2) 3 - (3 \log_2 3 + 1 - 3 \log_3 2) \approx 1.19 \ldots > 0.
\end{aligned}$$

(2): For $k \geqslant 2$ and $i \leqslant k-1$,

$$\begin{aligned}
&2K + 3 + \lceil \log_3 2^{i-1} \rceil + \lceil \log_3 (2^k - 2^i) \rceil \\
&\quad = (2K + 2 + \lceil \log_3 2^{i-1} \rceil) + (1 + \lceil \log_3 (2^k - 2^i) \rceil) \\
&\quad \leqslant f(2^k + 2^i, 2) + f(2^k - 2^i, 1) \quad \text{(by Lemmas 5.4(1) and 5.5(3))} \\
&\quad \leqslant f(2^{k+1}, 3).
\end{aligned}$$

(3): Holds by Lemma 5.3(1) iff it holds for $m = 2^k$, which is exactly (2).

(4): Holds by Lemma 5.3(2) iff it holds for $k = \lceil \log_2 m \rceil$. When $k = \lceil \log_2 m \rceil$,

$$\begin{aligned}
&K + 2 + k + \lfloor \lceil \log_3 \lfloor \tfrac{1}{2} l \rfloor \rceil + \lceil \log_3 (m - l) \rceil \\
&\quad = (2K + 2 + \lceil \log_3 \lfloor \tfrac{1}{2} l \rfloor \rceil) + (1 + \lceil \log_3 (m - l) \rceil) \\
&\quad \leqslant f(2^k + l, 2) + f(m - l, 1) \quad \text{(by Lemmas 5.4(1) and 5.5(8))} \\
&\quad \leqslant f(2^k + m, 3).
\end{aligned}$$

(5): Holds by Lemma 5.3(2) iff it holds for $k = \lceil \log_2 m \rceil$. When $k = \lceil \log_2 m \rceil$,

$$k + 2 + \lceil \log_2 m \rceil + \lceil \log_3 (m-2) \rceil = (2k+3) + \log_3(m-2)$$
$$\leqslant f(2^k + 2, 2) + f(m-2, 1) \quad \text{(By Lemma 5.4(1) and 5.4(3))}$$
$$\leqslant f(2^k + m, 3).$$

(6): Holds by Lemma 5.3(2) iff it holds for $k = \lceil \log_2 m \rceil$. When $k = \lceil \log_2 m \rceil$,

$$k + 3 + \lceil \log_2 m \rceil = 2k + 3 \leqslant f(2^k + 2, 2) \quad \text{(By Lemma 5.4(3))}$$
$$\leqslant f(2^k + m, 2) \leqslant f(2^k + m, m).$$

(7): Holds by Lemma 5.3(2) iff it holds for $k = \lceil \log_2 m \rceil$. When $k = \lceil \log_2 m \rceil$,

$$k + 3 + \lceil \log_2 m \rceil + \lceil \log_3 (m-2) \rceil = (2k+3) + (1 + \log_3(m-2))$$
$$\leqslant f(2^k + 2, 2) + f(m-2, 1) \quad \text{(By Lemmas 5.4(1) and 5.4(3))}$$
$$\leqslant f(2^k + m, 3) \leqslant f(2^k + m, m-1). \qquad \square$$

Lemma 7. (1) $1 + K + \lceil \log_2 (n - 2^k) \rceil \leqslant 2 \log_2 n$.

(2) $1 + \lceil \log_3 c \rceil + 2 \log_2 (n - c) \leqslant f(n, 2)$.

(3) $4 + \lceil \log_2 (n-4) \rceil \leqslant f(n, 2)$ for $f(n > 4$.

Proof. (1):

$$1 + K + \lceil \log_2 (n - 2^k) \rceil \leqslant 2 + k + \log_2 (n - 2^k) = 2 + \log_2 [2^k \cdot (n - 2^k)]$$
$$\leqslant 2 + \log_2 \left(\frac{n}{2} \cdot \frac{n}{2} \right) = 2 \log_2 n.$$

(2): Let $g(c) = f(n, 2) - [2 + \log_2 c + 2 \log_2 (n - c)]$. Then

$$g(c) = (2 + \log_3 2) \log_2 n - \log_3 c - 2 \log_2 (n - c) - (1 + \log_3 2)$$

and

$$g'(c) = \frac{1}{c \ln 3} - \frac{2}{(n-2) \ln 2}.$$

So

$$g'(c) = 0 \;\Leftrightarrow\; c = \frac{n}{1 + 2 \log_2 3}.$$

Hence

$$g(c) \geqslant g\left(\frac{n}{1 + 2 \log_2 3} \right) = \log_3(\tfrac{1}{2} + \log_2 3) + \log_2(1 + \tfrac{1}{2} \log_3 2) - 1 \approx 0.05 > 0.$$

Thus

$$1 + \lceil \log_3 c \rceil + 2\log_2 (n - c) \leqslant f(n, 2).$$

(3): If $4 < n < 8$, the inequality can be verified directly. For $n \geqslant 8$,

$$\begin{aligned} f(n, 2) - (4 + \lceil \log_2(n - 4) \rceil) &\geqslant (2 + \log_3 2)\log_2 \tfrac{1}{2} n + 3 - (5 + \log_2 (n - 4)) \\ &= (2 + \log_3 2)\log_2 n - (2 + \log_3 2) - (2 + \log_2 (n - 4)) \\ &= 2\log_2 n + \log_3 n - \log_2 (n - 4) - 4 - \log_3 2 \\ &= (\log_2 n - \log_2 (n - 4)) + (\log_2 n - 3) + (\log_3 n - \log_3 2 - 1) \\ &> 0 + 0 + 0 = 0. \end{aligned}$$

Thus for all $n > 4$,

$$4 + \lceil \log_2 (n - 4) \rceil \leqslant f(n, 2). \qquad \square$$

References

[1] S.S. Cairns, Balance scale sorting, Amer. Math. Monthly 70 (1963) 136–148.

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

[3] D.-Z. Du and H. Park, On Competitive Group Testing, SIAM J. Comput. 23(5) (1994) 1019–1025.

[4] X.-D. Hu, P.-D. Chen and F.K. Hwang, A new competitive algorithm for the counterfeit coin problem, Inform. Process Lett. 51 (1994) 213–218.

[5] X.-D. Hu and F.K. Hwang, A competitive algorithm for the counterfeit coin problem, in: D.-Z. Du and P.M. Pardalos, eds., Minimax and Applications (Kluwer Academic Publishers, 1995) 241–250.