JMLR: Workshop and Conference Proceedings vol 30 (2013) 1–22

Polynomial Time Optimal Query Algorithms for Finding Graphs with Arbitrary Real Weights

Sung-Soon Choi SUNGSOONC@GMAIL.COM
Cadence Design Systems, Inc.

Abstract

We consider the problem of finding the edges of a hidden weighted graph and their weights by using a certain type of queries as few times as possible, with focusing on two types of queries with additive property. For a set of vertices, the additive query asks the sum of weights of the edges with both ends in the set. For a pair of disjoint sets of vertices, the cross-additive query asks the sum of weights of the edges crossing between the two sets. These queries are related to DNA sequencing and finding Fourier coefficients of pseudo-Boolean functions, and have been paid attention to in computational learning.

In this paper, we achieve an ultimate goal of recent years for graph finding, by constructing the first polynomial time algorithms with optimal query complexity for the general class of graphs with n vertices and at most m edges in which the weights of edges are arbitrary real numbers. The algorithms are randomized and their query complexities are $\mathcal{O}\left(\frac{m \log n}{\log m}\right)$ which improve the best known bounds by a factor of $\log m$.

To build a key component for graph finding, we consider coin weighing with a spring scale which itself has been paid attention to in a long history of combinatorial search. We construct the first polynomial time algorithm with optimal query complexity for the general case in which the weight differences between counterfeit and authentic coins are arbitrary real numbers. We also construct the first polynomial time optimal query algorithm for finding Fourier coefficients of a certain class of pseudo-Boolean functions.

Keywords: Graph finding, coin weighing, Fourier coefficients, pseudo-Boolean functions, query complexity

1. Introduction

Graph finding. In graph finding, we are given a graph of which vertices are known a priori but edges are unknown. The problem is to find the edges by using a certain type of queries on the edges, as few times as possible. This paper focuses on two types of queries with additive property, *additive* and *cross-additive queries*. For a set of vertices, the additive query asks the number of edges with both ends in the set, and for a pair of disjoint sets of vertices, the cross-additive query asks the number of edges crossing between the two sets. For weighted graphs, as in this paper, the queries are generalized to ask the sum of weights of the relevant edges and the problem is extended to finding the edges of a given graph and their weights. Our primary concern is to construct polynomial time algorithms with optimal query complexity for a significant class of weighted graphs.[1]

1. Hereafter, 'optimal' means the relevant bound matches a lower bound up to a constant factor.

Additive query was motivated by shotgun sequencing which is one of the most popular methods for DNA sequencing; a main process of shotgun sequencing is reduced to graph finding with additive queries (Grebinski and Kucherov, 1998; Beigel et al., 2001). Cross-additive query is related to finding Fourier coefficients of pseudo-Boolean functions (Choi and Kim, 2010; Choi et al., 2011a) on which there will be a discussion later. Graph finding for the two queries may be regarded as an extension of coin weighing with a spring scale. A discussion on coin weighing will also be given later. For other types of queries, see (Alon and Asodi, 2005; Alon et al., 2004; Angluin and Chen, 2004, 2006; Hein, 1989; King et al., 2003; Reyzin and Srivastava, 2007a,b).

Until a few years ago, most studies on additive queries were conducted for unweighted graphs (Grebinski and Kucherov, 1998, 2000; Grebinski, 1998; Reyzin and Srivastava, 2007a; Choi and Kim, 2008, 2010; Mazzawi, 2010). In recent years, there have been an increasing number of studies on weighted graphs. For weighted graphs with n vertices and at most m edges of real weights with a mild condition, Choi and Kim (2010) non-constructively proved a non-adaptive optimal $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ query bound, provided that m is at least a polylog of n. Bshouty and Mazzawi extended this to arbitrarily small m (2011b), to arbitrary real weights (2011a), and to constant-bounded hypergraphs (2010b). It was not clear from these results, however, how to design polynomial time algorithms with optimal query complexity. Thus, the design of such an algorithm, especially for the general class of graphs with arbitrary real weights, has been considered as an ultimate goal of recent years, at least for practical applications.

In this research direction, for graphs with positive real weights, Bshouty and Mazzawi (2010a) presented a deterministic polynomial time algorithm with near optimal $\mathcal{O}\big(\frac{m \log n}{\log m} + m \log \log m\big)$ query complexity. Then, Choi et al. (2011b) constructed a randomized polynomial time algorithm with optimal $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ query complexity. For graphs with integer weights of constant-bounded absolute values, they also gave a randomized polynomial time algorithm using $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ queries. Kim (2012) extended this to real weights of which absolute values are bounded below and above by positive constants. (These algorithms are all adaptive.)

In this paper, we give a construction of a randomized polynomial time algorithm using $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ queries for graphs with arbitrary real weights. This bound is optimal for all m by an information-theoretic lower bound argument. It improves the best known bound $\mathcal{O}(m \log n)$ which can be obtained from (Choi et al., 2011a) by a factor of $\log m$. To the best of our knowledge, this is the first polynomial time algorithm with optimal query complexity.

To get the result on additive queries, we rather consider cross-additive queries. Since a cross-additive query may be answered by three additive queries (see (Choi and Kim, 2010) for more detail), the result on additive queries immediately follows from the following.

Problem 1 (Graph finding) Input: a weighted graph G for which the only information given is that (i) G has the vertex set $\{1, \ldots, n\}$ and at most m edges, and (ii) the weights of edges of G are arbitrary non-zero real numbers. Output: the edges of G and their weights.

Theorem 1 *One can construct a randomized polynomial time adaptive algorithm which solves Problem 1 with probability* $1 - o(1)$ *by using* $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ *cross-additive queries.*

This bound is also optimal for all m, and improves the best known bound $\mathcal{O}(m \log n)$ from (Choi et al., 2011a) by a factor of $\log m$.

Coin weighing. To build a key component for graph finding, we consider coin weighing with a spring scale. In the problem, a set of coins are given among which there are some counterfeit coins. Each counterfeit coin has a different weight from the weight of authentic coins which is known a priori (and so may be assumed to be 0). The problem is to find the counterfeits by using a spring scale as few times as possible. Given any set of coins, the scale gives the total weight of the coins in the set.

Coin weighing has been paid attention to in a long history of combinatorial search. In the early stages, studies mainly focused on the 0/1 weight cases (i.e., the cases in which the weights of counterfeits are all the same) and some extensions to integer weights. See (Söderberg and Shapiro, 1963; Erdős and Rényi, 1963; Cantor, 1964; Lindström, 1964, 1965, 1971; Cantor and Mills, 1966; Moser, 1970) for the results on the simplest 0/1 case with an arbitrary number of counterfeits. For two other famous cases, the 0/1 case with bounded number of counterfeits, and the case of non-negative integer weights with bounded total weight, see (Lindström, 1975; Capetanakis, 1979a,b; Tsybakov and Mikhailov, 1978; Massey, 1981; Uehara et al., 2000; Du and Hwang, 1993; Grebinski and Kucherov, 2000; Bshouty, 2009) and (Pippenger, 1981; Ruszinkó and Vanroose, 1997; Grebinski and Kucherov, 2000; Bshouty, 2009), respectively.

A few recent studies shed light on optimal bounds for real weights. When there are at most m counterfeits among n coins and the weights of counterfeits are real numbers with a mild condition, Choi and Kim (2010) proved a non-adaptive optimal $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ query bound. Bshouty and Mazzawi (2011a) extended this to arbitrary real weights. These studies, however, did not cover the design of polynomial time algorithms. As a study toward polynomial time optimal query algorithms, for counterfeits of positive real weights, Bshouty and Mazzawi (2010a) constructed a deterministic polynomial time algorithm using $\mathcal{O}\big(\frac{m \log n}{\log m} + m \log \log m\big)$ coin weighings. Then, Choi et al. (2011b) constructed a randomized polynomial time algorithm with optimal $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ query complexity. For integer weights of constant-bounded absolute values, they also presented a randomized polynomial time algorithm of $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ query bound, and Kim (2012) extended this to real weights of which absolute values are lower- and upper-bounded by positive constants. (These algorithms are all adaptive.)

In this paper, for arbitrary real weights, we construct a randomized polynomial time algorithm using $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ weighings. This bound is optimal for all m by an information-theoretic argument, and improves the best known bound $\mathcal{O}(m \log \frac{n}{m})$ from some recent results in compressed sensing, e.g., those based on expander graphs (Berinde et al., 2008; Indyk and Ruzic, 2008; Gilbert and Indyk, 2010)[2], by a factor of $\log n$ if $n^{\varepsilon} \leq m \leq n^{1-\varepsilon}$ for a fixed constant $\varepsilon > 0$. To the best of our knowledge, this is the first polynomial time algorithm with optimal query complexity for the general case in the long history of coin weighing.

Problem 2 (Coin weighing) Input: a set of coins for which the only information given is that (i) the total number of coins is n and among them, there are at most m counterfeits,

2. Notice that not all the results for compressed sensing yield the same bounds for coin weighing due to the restriction in the measurements in coin weighing.

(ii) all of the authentic coins have the same weight which is known a priori, and (iii) the weight difference between each counterfeit and an authentic coin is an arbitrary non-zero real number. Output: the counterfeits and their weights.

Theorem 2 *One can construct a randomized polynomial time adaptive algorithm which solves Problem 2 with probability* $1 - o(1)$ *by using* $\mathcal{O}\left(\frac{m \log n}{\log m}\right)$ *coin weighings, where the error probability may be reduced to* $\mathcal{O}\left(\frac{1}{m^c}\right)$ *for any large constant* $c > 0$.

Finding Fourier coefficients of pseudo-Boolean functions. Cross-additive query was motivated by the problem of finding Fourier coefficients of *bounded pseudo-Boolean functions*. A pseudo-Boolean function is a real-valued function defined on the set of binary strings of fixed length. It is k-bounded if it can be expressed as a sum of subfunctions each depending on at most k input bits.[3] Finding the Fourier coefficients of a k-bounded function is reduced to a series of hypergraph finding problems with (generalized) cross-additive queries, e.g., see (Choi and Kim, 2010) for $k = 2$.

Finding Fourier coefficients of k-bounded functions has important meanings in research areas including evolutionary computation and population genetics. For more details, see (Choi and Kim, 2010). After some initial studies (Kargupta and Park, 2001; Heckendorn and Wright, 2004; Choi et al., 2009), Choi et al. (2008, 2011a) gave a randomized adaptive polynomial time algorithm with $\mathcal{O}(m \log n)$ query complexity for k-bounded functions with n input bits and at most m non-zero Fourier coefficients, for constant k. For the 2-bounded functions with a mild condition on Fourier coefficients, Choi and Kim (2010) non-constructively proved a non-adaptive optimal $\mathcal{O}\left(\frac{m \log n}{\log m}\right)$ query bound, provided that m is at least a polylog of n.

In this paper, we have a (first) polynomial time optimal query algorithm for 2-bounded functions with arbitrary real Fourier coefficients. This follows from Theorems 1 and 2 and the reduction from this problem to a combination of graph fining with cross-additive queries and coin weighing (Choi and Kim, 2010).

Problem 3 (Fourier coefficients) Input : a function f for which the only information given is that (i) f is a 2-bounded pseudo-Boolean function defined on $\{0,1\}^n$, (ii) f has at most m non-zero Fourier coefficients, and (iii) the non-zero Fourier coefficients of f are arbitrary real numbers. Output : the Fourier coefficients of f.

Theorem 3 *One can construct a randomized polynomial time adaptive algorithm which solves Problem 3 with probability* $1 - o(1)$ *by using* $\mathcal{O}\left(\frac{m \log n}{\log m}\right)$ *function evaluations.*

The rest of the paper is organized as follows. In the next section, described are some preliminary results for understanding our result. Then, we prove Theorem 2 (coin weighing) and Theorem 1 (graph finding) in Sections 3 and 4, respectively.

3. Thus by definition, a k-bounded (pseudo-Boolean) function is a multilinear polynomial of degree k or less defined on the hypercube, and vice versa.

2. Preliminaries

2.1. Fourier-Based Search Matrix

A *search matrix* is a 0/1 matrix which maps each 0/1 vector of fixed length to a different output vector. Bshouty (2009) introduced a method to construct a $2^\nu \times \nu 2^{\nu-1}$ search matrix for any integer $\nu \geq 1$, which is optimal in the number of rows, and to decode it efficiently based on Fourier analysis of a relevant Pseudo-Boolean function. The method may be abstracted as follows, for use in various settings of coin weighing. In the following, for a set S of coins, $w(S)$ is the weight of S, i.e., the sum of weights of all coins in S. Unless specified otherwise, log means the logarithm to base 2.

Proposition 4 (Bshouty, 2009) *Let $A_1, \ldots, A_q$ be disjoint sets of coins and let ν be the smallest integer satisfying $\nu 2^{\nu-1} \geq q$. Then, one can use 2^ν coin weighings in a non-adaptive way to find the following in polynomial time in q: For each $i = 1, \ldots, q$, real numbers $x_i, a_{ik}, k = 1, \ldots, i-1$ and an integer ℓ_i with $0 \leq \ell_i < \nu$ such that*

$$w(A_i) = x_i - \sum_{k=1}^{i-1} a_{ik} w(A_k) - \sum_{k=1}^{\ell_i} \frac{w(A_{i+k})}{2^k}.$$

In particular, the number 2^ν of weighings is at most $\frac{(4+o(1))q}{\log q}$, where $o(1)$ approaches to 0 as q increases.

For example, it is straightforward to use the proposition to get a polynomial time optimal query algorithm for the simplest 0/1 weight case: Let $c_1, \ldots, c_n$ be n given coins. Regarding each c_i as a set, inductively on $i = 1, \ldots, n$, use the proposition to find the value $x_i - \sum_{k=1}^{i-1} a_{ik} w(c_k)$. Then, we see $w(c_i) = \lfloor x_i - \sum_{k=1}^{i-1} a_{ik} w(c_k) \rfloor$ as $0 \leq \sum_{k=1}^{\ell_i} \frac{w(c_{i+k})}{2^k} < \sum_{k=1}^{\infty} \frac{1}{2^k} = 1$. By the proposition (with $q = n$) again, the number of weighings is $2^\nu = \mathcal{O}\left(\frac{n}{\log n}\right)$, which is optimal.

2.2. Guess-and-Fix Strategy

Many recent algorithms for coin weighing (Bshouty, 2009; Bshouty and Mazzawi, 2010a; Choi et al., 2011b) are advanced forms of a simple algorithm called the *divide-and-weigh* for non-negative real weights. Initially in the algorithm, the set of all given coins is weighed. If the weight is zero, the algorithm is terminated, and otherwise, let $\mathcal{S}_0$ be the collection containing only the set. Letting n be the number of coins, incrementally on $t = 1, \ldots, \lceil \log n \rceil$, divide each $S_i \in \mathcal{S}_{t-1}$ into two subsets S_{i1} and S_{i2} of equal size (up to one), and weigh S_{i1}'s one by one and accordingly, find the weights of S_{i2}'s by $w(S_{i2}) = w(S_i) - w(S_{i1})$. Then, let $\mathcal{S}_t$ be the collection of S_{ij}'s $j = 1, 2$ with non-zero weights. Since each set in $\mathcal{S}_{\lceil \log n \rceil}$ contains only one counterfeit, all counterfeits and their weights are found from the sets in $\mathcal{S}_{\lceil \log n \rceil}$. The query complexity of this algorithm is $\mathcal{O}\left(m \log \frac{n}{m}\right)$ when the number of counterfeits is at most m. See (Uehara et al., 2000) for a proof idea.

A *guess-and-fix* strategy was proposed in the Bshouty-Mazzawi (BM) algorithm (Bshouty and Mazzawi, 2010a) to find the weights of subsets obtained by division in each iteration of the divide-and-weigh, overall more efficiently. To this end, it iteratively chooses a group of subsets, guesses their weights, checks the correctness of the guesses, and fixes incorrect ones (if exist). More specifically, in the t^{th} iteration, let $\mathcal{S}_{t-1} = \{S_1, \ldots, S_q\}$ and w.l.o.g.

assume that S_i's are in a non-increasing order of the weights. To find $w(S_{ij})$'s $j = 1, 2$, an (integer) interval $I \subseteq \{1, \ldots, q\}$ starting from 1 is initially chosen. Then, $w(S_{i1})$'s $i \in I$ are guessed sequentially, assuming that each $S_i, i \in I$ is not *split*, i.e., the counterfeits in S_i are inherited to only one of S_{i1} and S_{i2}. Under the assumption, each $w(S_{i1})$ $i \in I$ must be one of the two values, $w(S_i)$ and 0. Hence, as in the above 0/1 weight case, Proposition 4 is used to guess $w(S_{i1})$'s $i \in I$ with $\mathcal{O}\big(\frac{q}{\log q}\big)$ weighings as follows:

Inductively on $i = 1, \ldots, |I|$, the guess for $w(S_{i1})$, say $u(S_{i1})$ is set to be $w(S_i)$ if $x_i - \sum_{k=1}^{i-1} a_{ik} u(S_{k1}) \geq w(S_i)$, and it is set to be 0 otherwise. It is not hard to see that if S_i's are not split for all $i \in I$, then $u(S_{i1})$'s are correct, i.e., $u(S_{i1}) = w(S_{i1})$ for all $i \in I$, from the fact that $0 \leq \sum_{k=1}^{\ell_i} \frac{w(S_{i+k,1})}{2^k} \leq \sum_{k=1}^{\ell_i} \frac{w(S_{i+k})}{2^k} \leq \sum_{k=1}^{\ell_i} \frac{w(S_i)}{2^k} < w(S_i)$. After getting $u(S_{i1})$ $i \in I$, the weights of S_{i2} $i \in I$ are guessed to be $u(S_{i2}) = w(S_i) - u(S_{i1})$. We will call this the *guess* operation for the interval I.

Then, the *check-and-fix* operation for I follows: First, it is checked whether $u(S_{ij})$ $i \in I$ $j = 1, 2$ are all correct, simply by weighing all the sets with 0-guesses (S_{ij}'s with $u(S_{ij}) = 0$) together to see whether the total weight is 0. If the 0-guesses are all correct, which means that $w(S_{ij})$ $i \in I$ $j = 1, 2$ are all found, then choose an interval I' next to I. (I' starts from $|I| + 1$.) Otherwise, find the first incorrect 0-guess via binary search and fix the corresponding pair, say $u(S_{i^*j})$ $j = 1, 2$. Then, choose an interval I' next to the fixed pair. (I' starts from $i^* + 1$.) Now, the guess and check-and-fix operations are repeated for I'.

The efficiency of the guess-and-fix strategy comes from the guess operations with optimal query complexity. By the proposition, once the $\mathcal{O}\big(\frac{q}{\log q}\big)$ weighings are conducted for the initial interval, the guess operations for the other intervals may be conducted without additional weighings. Hence, the BM algorithm uses $\mathcal{O}\big(\frac{m}{\log m}\big)$ weighings for guess operations in each iteration, as the size $|\mathcal{S}_t|$ of $\mathcal{S}_t$ is at most m for each t. On the other hand, it uses a fixed interval size $\Theta(\log m)$ (except in few cases), and so $\mathcal{O}\big(\frac{m}{\log m}\big)$ weighings are required for check operations (not for fix operations) in each iteration. From these, the first term in its query complexity $\mathcal{O}\big(\frac{m \log n}{\log m} + m \log\log m\big)$ follows. The second term comes from the facts that $\mathcal{O}(\log\log m)$ weighings are used (for binary search) for each fix operation, and the number of guesses to be fixed over all iterations is $\mathcal{O}(m)$ (as it is at most twice the number of sets split during the algorithm execution).

The algorithm of (Choi et al., 2011b) uses a random division of coin sets so that with high probability (w.h.p.) the number of sets split in the t^{th} iteration is bounded above by exponentially decreasing values in t for enough number of t's in the beginning. According to this, it uses an adaptive interval size 2^t in the t^{th} iteration. These features yield exponentially decreasing costs for fix operations, over a number of iterations in the beginning, to eliminate the second term in the query complexity of the BM algorithm.

3. Coin Weighing Algorithm

3.1. Algorithm Overview and Main Ideas

Allowing negative weights brings a few obstacles to directly applying the algorithms for non-negative weights. At first, a set of weight 0 may contain two or more counterfeits of which weights cancel out one another, and so we may not always discard the coins in the set from further consideration. To overcome this, we use an idea in (Choi et al., 2011b)

for bounded integer weights. Our algorithm consists of two phases. In the first phase, it iterates $\Theta(\log\log m)$ rounds in each of which, whenever a set of weight 0 is found, it keeps the coins in the set aside to examine them again in the next rounds (or in the next phase). In each round, among the counterfeits not found so far, the algorithm finds a portion of the counterfeits, excluding at most $\frac{2}{3}$ portion of them. After the first phase, there remain $\mathcal{O}\big(\frac{m}{\log m}\big)$ counterfeits not found. In the second phase, the algorithm finds them one by one by a randomized binary search which uses $\mathcal{O}(\log n)$ weighings per counterfeit.

In each round (of the first phase), an upper bound q is given on the number of counterfeits not found so far. (Specifically, q is set to be $m\big(\frac{2}{3}\big)^{r-1}$ in the r^{th} round for $r = 1, 2, \ldots$.) Then, the algorithm finds counterfeits with $\mathcal{O}\big(\frac{q\log n}{\log q}\big)$ weighings, leaving behind at most $\frac{2}{3}q$ ones. To this end, the coins not found so far are initially distributed into q sets: Each coin is put into one of q sets uniformly at random (u.a.r.) and independently of other coins. The q sets are weighed one by one, and only the sets of non-zero weights are put into $\mathcal{S}_0$. Now, incrementally on $t = 1, 2, \ldots$, in the t^{th} iteration, the algorithm divides each $S_i \in \mathcal{S}_{t-1}$ into two subsets S_{ij} $j = 1, 2$ such that each coin in S_i is independently put into one of S_{ij}, each with probability $\frac{1}{2}$. Then, it finds $w(S_{ij})$'s based on a new type of guess-and-fix strategy, which we will present below, and puts only the sets of non-zero weights into $\mathcal{S}_t$. After $\Theta(\log n)$ iterations, only the sets of size one remain, and the coins therein are the counterfeits to be found in the round.

It turns out that counterfeit coins except at most $\frac{2}{3}q$ ones are isolated in the initial distribution, and so the number of counterfeits not found in the round is at most $\frac{2}{3}q$. Also, the query complexity for the guess-and-fix strategy is upper-bounded by an exponentially decreasing sequence starting from $\mathcal{O}\big(\frac{q\log n}{\log q}\big)$ for an enough number of initial iterations, from which a desired query complexity follows for the round.

Randomized guess-and-fix. For sets $S_i \in \mathcal{S}_{t-1}$ and their subsets S_{ij} $j = 1, 2$ in the t^{th} iteration, we propose a randomized guess-and-fix strategy to find $w(S_{ij})$'s of which query complexity is properly bounded.

At first, we give a new construction scheme of search matrices which will be used for sets of coins with arbitrary real weights. The following abstracts the essence of the scheme, for coin weighing. Hereafter, for positive integers x and y, we define $L(x, y) := \sum_{j=0}^{x} \min(j, y)\binom{x}{j}$.

Lemma 5 *Let $A_1, \ldots, A_q$ be disjoint sets of coins. Let y be a positive integer and let ν be the smallest positive integer satisfying $L(\nu, y) \geq 2q$. Then, one can use 2^ν coin weighings in a non-adaptive way to find the following in polynomial time in q: For each $i = 1, \ldots, q$, real numbers $x_i, a_{ik}, k = 1, \ldots, i-1$ and an integer ℓ_i with $0 \leq \ell_i < \min(\nu, y)/2$ such that*

$$w(A_i) = x_i - \sum_{k=1}^{i-1} a_{ik} w(A_k) - \sum_{k=1}^{\ell_i} \frac{w(A_{i+k})}{4^k}.$$

In particular, for a universal constant $\alpha > 0$, the number 2^ν of weighings is at most $\big(1 + \frac{\alpha}{\sqrt{\log 2q}}\big) \max\Big(\frac{4q}{y}, \frac{8q}{\log 2q}\Big)$ for sufficiently large q.

Proof See Appendix A. ■

As seen in the below, we use different y's depending on t to control a tradeoff between the

number of weighings for constructing equations for guessing and an upper bound on the number of certain incorrect guesses.

Lemma 5 is used for guessing $w(S_{ij})$'s as follows. W.l.o.g., assume that S_i's are labeled in a non-increasing order of $|w(S_i)|$'s. Writing $q_t := |\mathcal{S}_{t-1}|$, use the lemma with $A_i = S_{i1}$ $i = 1, \ldots, q_t$ (and a positive integer y) to get the equations for $w(S_{i1})$'s. Given $w(S_i)$'s, we guess $w(S_{i1})$'s inductively on $i = 1, \ldots, q_t$: The guess $u(S_{i1})$ for $w(S_{i1})$ is set to be $w(S_i)$ if $|x_i - \sum_{k=1}^{i-1} a_{ik} u(S_{k1})| \geq |w(S_i)|/2$, and it is set to be 0 otherwise. The guess for $w(S_{i2})$ is set to be $w(S_i) - u(S_{i1})$.

If S_i is not split for all i, then $u(S_{ij})$'s are correct for all i, as $|\sum_{k=1}^{\ell_i} \frac{w(S_{i+k,1})}{4^k}| \leq \sum_{k=1}^{\ell_i} \frac{|w(S_{i+k})|}{4^k} \leq \sum_{k=1}^{\ell_i} \frac{|w(S_i)|}{4^k} < |w(S_i)|/2$. For an interval $I \subseteq [1, \ldots, q_t]$, we may guess $w(S_{i1})$'s for $i \in I$ in the same way if $w(S_i)$'s for $i \in I$ and $w(S_{k1})$'s for $k < \min(I)$ are given.

Though a guessing operation is now available for arbitrary real weights, there are two problems which should be addressed when we apply the (deterministic) guess-and-fix strategies for non-negative real weights to arbitrary real weights.

(i) *Ambiguity in the meaning of weight zero*: The task of checking whether a group of sets are all of weight 0 plays a crucial role in the check-and-fix operations of the previous strategies. Unlike for non-negative weights, for conducting the task, it is not always enough to weigh the sets together to see whether the total weight is 0 (similarly to when checking whether a group of coins are all authentic, mentioned above).

(ii) *Unboundedness of the weights of the sets obtained by division*: Unlike for non-negative weights, $w(S_{i1})$ may be arbitrarily larger than $w(S_i)$ in absolute value. Thus, the guess for $w(S_{i1})$, based on the equation in Lemma 5, may be incorrect even when S_i is not split and the guesses for $w(S_{k1})$ with $k < i$ are correctly given, as the term $\sum_{k=1}^{\ell_i} \frac{w(S_{i+k,1})}{4^k}$ may be arbitrarily large in absolute value.

For the checking task in (i), it seems natural to consider a random testing procedure as follows.

---

procedure CHECK-ALL-ZEROS ($\mathcal{S}$: a group of coin sets, K: a positive integer)
  repeat K **times**
    generate a uniform random subset $\mathcal{S}'$ of $\mathcal{S}$
    weigh the union of the sets in $\mathcal{S}'$ to find its weight $w_{\mathcal{S}'}$
    if $w_{\mathcal{S}'} \neq 0$, **return** FALSE
  return TRUE

For making the existing check-and-fix operations work for arbitrary real weights, it might be an initial try to use CHECK-ALL-ZEROS (with proper K) for each checking task in the operations and bound the overall error probability by union bound. It turns out, however, that over the operations, the task is to be conducted too often, say $\Omega(m)$ times. Thus, K should be $\Omega(\log m)$ to use union bound, which yields a non-optimal query complexity.

Our strategy to overcome (i) and (ii) is to allow some constant error probability for each checking task, but also to check the correctness of previously conducted tasks on a regular basis. More specifically in the t^{th} iteration, we choose intervals $I \subseteq [1, \ldots, q_t]$ of size 2^t (or less in few cases) and guess $w(S_{i1})$'s for $i \in I$. For small constant $\varepsilon > 0$, we use CHECK-ALL-ZEROS with $K \geq \log(\frac{t+1}{\varepsilon})$ inside the check-and-fix operations so as to

get an error probability at most ε for each check-and-fix operation. In addition, before starting the guess/check-and-fix operation for an interval I (except the one starting 1), we check the correctness of the all 0-guesses for the sets located before I by using Check-All-Zeros with $K \geq \log\left(\frac{1}{\varepsilon}\right)$. The error probability for this is also at most ε. If the procedure reveals the existence of an incorrect 0-guess before I, then the most recent guess/check-and-fix operations (for the last interval) are *rolled back*. Otherwise, the guess/check-and-fix operation is conducted for the interval I.

As will be seen, the whole guess-and-fix process to find the weights of sets obtained by division in an iteration, may be regarded as a series of transitions on a Markov chain (MC) with a unique absorbing state in which each transition corresponds to a guess/check-and-fix operation or a roll-back operation. In particular, by setting K appropriately for ε as above, each transition going one step closer to the absorbing state occurs with probability at least $1-\varepsilon$. Thus, based on a large deviation inequality for a sum of (geometric-like) random variables, the number of weighings for the check-and-fix and roll-back operations is w.h.p. within a constant factor of the distance between the starting and absorbing states in the MC.

The distance in the MC depends on the number of *leading incorrect guesses* induced by the equations from a search matrix which is used for guessing in the iteration. Suppose that in the t^{th} iteration, let $\mathcal{S}_{t-1} = \{S_1, \ldots, S_{q_t}\}$ where $|w(S_1)| \geq \cdots \geq |w(S_{q_t})|$, and we get the equations from Lemma 5 with $A_i = S_{i1}$ $i = 1, \ldots, q_t$ to guess $w(S_{i1})$'s. Then, the guess for $w(S_{i1})$, based on the equation for $w(S_{i1})$, is called a leading incorrect guess if it is incorrect when the guesses for $w(S_{k1})$ $k < i$ are correctly given, along with the value of $w(S_i)$.

We should notice that for each S_k, there are $\max_{1\leq j\leq q_t} \ell_j$ or less i's with $i < k$ such that $w(S_{i1})$ depends on $w(S_{k1})$ in the equation for $w(S_{i1})$ in the lemma. Thus, the number of leading incorrect guesses in the iteration is at most $(1 + \max_{1\leq j\leq q_t} \ell_j)$ times the number of S_k's split. Since $\max_{1\leq j\leq q_t} \ell_j < y$, we may set y appropriately to bound the number of leading incorrect guesses. We will set $y = \lceil 1.1^t \rceil$ in the t^{th} iteration. It turns out that for an upper bound q on the number of counterfeits in a round, the number of sets split in the t^{th} iteration of the round is $\mathcal{O}(\frac{q}{2^t} + q^{\frac{3}{4}})$ (w.h.p.). Hence, we get an exponentially decreasing upper bound on the number of leading incorrect guesses, and consequently, an exponentially decreasing upper bound on the distance of the MC, in the t^{th} iteration for an enough number of small t's.

In this way, it is possible to get a proper bound on the number of weighings for check-and-fix and roll-back operations in each iteration. The number of weighings for constructing equations for guessing is also properly bounded.

3.2. Algorithm Description

We present the pseudocode of the algorithm. It consists of a main procedure, Find-Counterfeits, and five subprocedures (including Check-All-Zeros presented in Section 3.1). The main procedure takes a set $\mathcal{C}$ of n coins with at most m counterfeits, and returns the set $\mathcal{C}_{\text{ct}}$ of counterfeits and their weights $(u(c))_{c\in\mathcal{C}_{\text{ct}}}$ found by the algorithm. (In the pseudocode, for a set S of coins, $u(S)$ is used to represent the weight of S computed by the algorithm. As before, $w(S)$ represents the real weight of S.) The values of n and m

may be accessed by any procedure in the pseudocode. All other variables are local to the procedures in which they appear.

In the main procedure, the variables $\mathcal{C}_{\rm ct}$ and $\mathcal{C}_{\rm un}$ contain the counterfeits found so far and the coins unknown so far, respectively. The **for** and **repeat** loops correspond to the first and second phases of the algorithm. The $r^{\rm th}$ iteration of the **for** loop implements the $r^{\rm th}$ round of the first phase, and calls Find-Portion-of-Counterfeits with an upper bound $m(\frac{2}{3})^{r-1}$ on the number of counterfeits in $\mathcal{C}_{\rm un}$ to find some of the coins in $\mathcal{C}_{\rm un}$ and their weights. The **repeat** loop iteratively calls Find-One-Counterfeit which finds a counterfeit in $\mathcal{C}_{\rm un}$ by a randomized binary search, until there is no counterfeit left in $\mathcal{C}_{\rm un}$.

The procedure Find-Portion-of-Counterfeits implements each round in the first phase of the algorithm. Given a set $\mathcal{C}$ of coins and a positive integer q, it finds a portion of the counterfeits in $\mathcal{C}$, starting with the random distribution of the coins in $\mathcal{C}$ into q sets. For $t \le \lceil 2\log q \rceil$, the $t^{\rm th}$ iteration of the **for** loop implements the randomized guess-and-fix strategy to find $w(S_{ij})$ $j = 1, 2$ for each $S_i \in \mathcal{S}_{t-1}$ by using Check-All-Zeros, Guess, and Find-First-Incorrect. In particular, the **if** block of lines 11–14 implements a roll-back operation (with checking previous 0-guesses). The **if** block of lines 15–28 implements a guess/check-and-fix operation. For $t > \lceil 2\log q \rceil$, roll-back and check-and-fix operations are not conducted in the $t^{\rm th}$ iteration of the **for** loop. For integers a, b with $a \le b$, the notation $[a, b]$ means the integer interval between a and b. For an integer interval I, $\mathrm{pre}(I)$ denotes the integer interval between 1 and $\min(I) - 1$.

The procedure Guess computes the guesses for $w(S_{ij})$ $i \in I, j = 1, 2$, based on the equations for $w(S_{i1})$ $i \in I$ in Lemma 5 and the values $\big(u(S_i)\big)_{i\in I}$ and $\big(u(S_{i1})\big)_{i\in \mathrm{pre}(I)}$.

---

procedure Find-Counterfeits ($\mathcal{C}$: a set of n coins with at most m counterfeits)
  $\mathcal{C}_{\rm ct} \leftarrow \emptyset$ and $\mathcal{C}_{\rm un} \leftarrow \mathcal{C}$
  for r **from** 1 **to** $\lceil \log\log m \rceil$
    $\big(\mathcal{C}_r, (u(c))_{c\in\mathcal{C}_r}\big) \leftarrow$ Find-Portion-of-Counterfeits$\big(\mathcal{C}_{\rm un}, m\big(\frac{2}{3}\big)^{r-1}\big)$
    $\mathcal{C}_{\rm ct} \leftarrow \mathcal{C}_{\rm ct} \cup \mathcal{C}_r$ and $\mathcal{C}_{\rm un} \leftarrow \mathcal{C}_{\rm un} \setminus \mathcal{C}_r$
  repeat
    $\big(c, u(c)\big) \leftarrow$ Find-One-Counterfeit$(\mathcal{C}_{\rm un})$
    if $\big(c, u(c)\big) =$ Null, **exit**
    $\mathcal{C}_{\rm ct} \leftarrow \mathcal{C}_{\rm ct} \cup \{c\}$ and $\mathcal{C}_{\rm un} \leftarrow \mathcal{C}_{\rm un} \setminus \{c\}$
  return $\big(\mathcal{C}_{\rm ct}, (u(c))_{c\in\mathcal{C}_{\rm ct}}\big)$

---

procedure Find-One-Counterfeit ($\mathcal{C}$: a set of coins)
  repeat $\lceil 2\log m \rceil$ **times**
    generate a uniform random subset S of $\mathcal{C}$, and weigh S to find its weight w_S
    if $w_S \neq 0$,
      find a counterfeit c in S and its weight $w(c)$ by a deterministic binary search
      return $(c, w(c))$
  return Null

---

procedure Find-Portion-of-Counterfeits ($\mathcal{C}$: a set of n coins, q: a positive integer)
// ε is fixed to be $\frac{1}{16}$
put each coin in $\mathcal{C}$ into one of q (initially empty) sets u.a.r. and independently
weigh each of the q sets, say S, and if $w(S) \neq 0$, put S into $\mathcal{S}_0$ and $u(S) \leftarrow w(S)$
for t **from** 1 **to** $\lceil 2\log q \rceil + \lceil \log n \rceil$
  let $q_t := |\mathcal{S}_{t-1}|$, and **if** $q_t = 0$, **return** $(\emptyset, \text{Null})$
  label the sets in $\mathcal{S}_{t-1}$, say $S_1, \dots, S_{q_t}$ so that $|u(S_1)| \geq \cdots \geq |u(S_{q_t})|$
  if $t \leq \lceil 2\log q \rceil$
    divide each $S_i \in \mathcal{S}_{t-1}$ into S_{i1} and S_{i2} u.a.r. and independently of other sets
    use Lemma 5 with $A_i = S_{i1}, i = 1, \dots, q_t$ and $y = \lceil 1.1^t \rceil$ to find $\big(x_i, (a_{ik})_{k<i}\big)_{i\in[1,q_t]}$ for $w(S_{i1})$'s
    $I_1 \leftarrow \big[1, \min(2^t, q_t)\big]$ and $h \leftarrow 1$
    repeat $\lceil 9\big(\frac{q}{2^t} + q^{\frac{3}{4}}\big)\min(\lceil 1.1^t \rceil, \log q)\rceil$ **times**
      if $I_h = \emptyset$ or $\min(I_h) > 1$
        $\mathcal{U} \leftarrow \big\{S_{ij} : u(S_{ij}) = 0, i \in \text{pre}(I_h), j = 1, 2\big\}$
        if Check-All-Zeros$\big(\mathcal{U}, \lceil \log \frac{1}{\varepsilon} \rceil\big) \neq$ True
          $h \leftarrow h - 1$ and skip the rest of the current iteration of the **repeat** loop
      if $I_h \neq \emptyset$
        if $\min(I_h) = 1$
          $\big(u(S_{ij})\big)_{i\in I_h, j=1,2} \leftarrow$ Guess$\big(I_h, \big(u(S_i), x_i, (a_{ik})_{k<i}\big)_{i\in I_h}, \text{Null}\big)$
        else
          $\big(u(S_{ij})\big)_{i\in I_h, j=1,2} \leftarrow$ Guess$\big(I_h, \big(u(S_i), x_i, (a_{ik})_{k<i}\big)_{i\in I_h}, \big(u(S_{i1})\big)_{i\in \text{pre}(I_h)}\big)$
        $\mathcal{U} \leftarrow \big\{S_{ij} : u(S_{ij}) = 0, i \in I_h, j = 1, 2\big\}$
        if Check-All-Zeros$\big(\mathcal{U}, \lceil \log \frac{t+1}{\varepsilon} \rceil\big) =$ True
          if $\max(I_h) = q_t$, $I_{h+1} \leftarrow \emptyset$
          else $I_{h+1} \leftarrow \big[\max(I_h) + 1, \min\big(\max(I_h) + 1 + 2^t, q_t\big)\big]$
        else
          $(i^*, j^*, w_{i^*j^*}) \leftarrow$ Find-First-Incorrect$(\mathcal{U}, t)$
          $u(S_{i^*j^*}) \leftarrow w_{i^*j^*}$ and $u(S_{i^*\overline{j^*}}) \leftarrow u(S_{i^*}) - w_{i^*j^*}$ for $\overline{j^*} := j^* + 1 \pmod 2$
          if $i^* = q_t$, $I_{h+1} \leftarrow \emptyset$ **else** $I_{h+1} \leftarrow \big[i^* + 1, \min(i^* + 1 + 2^t, q_t)\big]$
        $h \leftarrow h + 1$
  else
    divide each $S_i \in \mathcal{S}_{t-1}$ into S_{i1} and S_{i2} of equal size (up to one) in a deterministic way
    use Lemma 5 with $A_i = S_{i1}, i = 1, \dots, q_t$ and $y = \lceil 1.1^t \rceil$ to find $\big(x_i, (a_{ik})_{k<i}\big)_{i\in[1,q_t]}$ for $w(S_{i1})$'s
    $\big(u(S_{ij})\big)_{i\in[1,q_t], j=1,2} \leftarrow$ Guess$\big([1, q_t], \big(u(S_i), x_i, (a_{ik})_{k<i}\big)_{i\in[1,q_t]}, \text{Null}\big)$
  $\mathcal{S}_t \leftarrow \big\{S_{ij} : u(S_{ij}) \neq 0, i \in [1, q_t], j = 1, 2\big\}$
$\mathcal{C}_{\text{fd}} \leftarrow \bigcup_{S\in\mathcal{S}_{\lceil 2\log q\rceil + \lceil \log n\rceil}} S$, and for each $S \in \mathcal{S}_{\lceil 2\log q\rceil + \lceil \log n\rceil}$, $u(c) \leftarrow u(S)$ for $c \in S$
return $\big(\mathcal{C}_{\text{fd}}, (u(c))_{c\in\mathcal{C}_{\text{fd}}}\big)$

---

procedure GUESS $\big(I$: an integer interval, $\big(u(S_i), x_i, (a_{ik})_{k<i}\big)_{i\in I}$, $\big(u(S_{i1})\big)_{i\in \mathrm{pre}(I)}\big)$
  for i **from** $\min(I)$ **to** $\max(I)$
    $s \leftarrow x_i - \sum_{k=1}^{i-1} a_{ik} u(S_{k1})$
    if $|s| \geq |u(S_i)|/2$, $u(S_{i1}) \leftarrow u(S_i)$ and $u(S_{i2}) \leftarrow 0$ **else** $u(S_{i1}) \leftarrow 0$ and $u(S_{i2}) \leftarrow u(S_i)$
  return $\big(u(S_{ij})\big)_{i\in I, j=1,2}$

---

procedure FIND-FIRST-INCORRECT ($\mathcal{U}$: a group of sets S_{ij}, t: a positive integer)
// ε is fixed to be $\frac{1}{16}$
  $\mathcal{U}_0 \leftarrow \mathcal{U}$
  while $|\mathcal{U}_0| > 1$
    $\mathcal{U}_1 \leftarrow \big\{S_{ij} : i$ is in the first $\lceil \frac{|\mathcal{U}_0|}{2} \rceil$ among the first indices of the sets in $\mathcal{U}_0\big\}$
    $\mathcal{U}_2 \leftarrow \mathcal{U}_0 \setminus \mathcal{U}_1$
    if CHECK-ALL-ZEROS$\big(\mathcal{U}_1, \lceil \log \frac{t+1}{\varepsilon} \rceil\big) \neq$ TRUE, $\mathcal{U}_0 \leftarrow \mathcal{U}_1$ **else** $\mathcal{U}_0 \leftarrow \mathcal{U}_2$
  weigh the set $S_{i^*j^*}$ in $\mathcal{U}_0$ to find its weight $w_{i^*j^*}$, and **return** $(i^*, j^*, w_{i^*j^*})$

3.3. Analysis

We analyze the error bound and the query complexity of the proposed algorithm. It is clear that the algorithm is of polynomial time. We first present a lemma which will be used for showing both the error bound and the query complexity.

Lemma 6 *Suppose that $\mathcal{C}$ is a set of coins among which there are at most q counterfeits. Then, with probability $1 - \mathcal{O}\big(\frac{1}{q}\big)$,* FIND-PORTION-OF-COUNTERFEITS$(\mathcal{C},q)$ *returns a set of counterfeits in $\mathcal{C}$ along with their weights which excludes at most $\frac{2}{3}q$ counterfeits in $\mathcal{C}$.*

Proof See Appendix B. ■

Error bound. The following shows the error bound of the algorithm.

Lemma 7 *Suppose that $\mathcal{C}$ is a set of coins among which there are at most m counterfeits. Then, with probability $1 - \mathcal{O}\big(\frac{\log m}{m}\big)$,* FIND-COUNTERFEITS$(\mathcal{C})$ *returns the set of counterfeits in $\mathcal{C}$ and their weights.*

Proof Inside the call FIND-COUNTERFEITS$(\mathcal{C})$, the overall error probability in the **for** loop is $\mathcal{O}\left(\sum_{r=1}^{\lceil \log \log m \rceil} 1/\left(\left(\frac{2}{3}\right)^{r-1} m\right)\right) = \mathcal{O}\big(\frac{\log m}{m}\big)$ by Lemma 6. On the other hand, the error probability in each iteration of the **repeat** loop is clearly $\mathcal{O}\left(\frac{1}{m^2}\right)$ and so the overall error probability in the **repeat** loop is $\mathcal{O}\left(\frac{1}{m}\right)$. From these, the lemma follows. ■

We should remark that the error bound may be further reduced to $\mathcal{O}\big(\frac{1}{m^c}\big)$ for arbitrarily large constant $c > 0$ if we increase by constant factors the number $\lceil 2 \log q \rceil$ in the number of iterations of the **for** loop in FIND-PORTION-OF-COUNTERFEITS as well as the number $\lceil 2 \log m \rceil$ of iterations of the **repeat** loop in FIND-ONE-COUNTERFEIT. As seen in the below, such a modification does not affect the (asymptotic) query complexity.

Query complexity. We first prove the following by using Lemma 5.

Lemma 8 *Suppose that $\mathcal{C}$ is a set of n coins among which there are at most q counterfeits. Then, the number of coin weighings used in* Find-Portion-of-Counterfeits($\mathcal{C}$,q) *is* $\mathcal{O}\big(\frac{q \log n}{\log q}\big)$.

Proof We first consider the number of weighings for constructing equations for guessing. By Lemma 5, for a universal constant $\alpha > 0$, the number of weighings is at most $\big(1 + \frac{\alpha}{\sqrt{\log 2q}}\big) \sum_{t=1}^{\lceil 2 \log q \rceil + \lceil \log n \rceil} \max\big(\frac{4q}{\lceil 1.1^t \rceil}, \frac{8q}{\log 2q}\big)$. Since $\sum_{t=1}^{\lceil 2 \log q \rceil + \lceil \log n \rceil} \frac{4q}{\lceil 1.1^t \rceil} \le \sum_{t=1}^{\infty} \frac{4q}{1.1^t} = \mathcal{O}(q)$ and $\sum_{t=1}^{\lceil 2 \log q \rceil + \lceil \log n \rceil} \frac{8q}{\log 2q} = \mathcal{O}\big(\frac{q \log n}{\log q}\big)$, the number of weighings is $\mathcal{O}\big(\frac{q \log n}{\log q}\big)$.

Next, consider the number of weighings for roll-back and check-and-fix operations. For each t with $1 \le t \le \lceil 2 \log q \rceil$, at most $\log 16 + (t+1)\lceil \log (16(t+1)) \rceil = 4 + (t+1)\lceil \log (16(t+1)) \rceil$ weighings are used in each iteration of the **repeat** loop, in the t^{th} iteration of the **for** loop. Hence, the total number is at most $\sum_{t=1}^{\lceil 2 \log q \rceil} \lceil 9(\frac{q}{2^t} + q^{\frac{3}{4}}) \min(\lceil 1.1^t \rceil, \log q) \rceil (4 + (t+1)\lceil \log (16(t+1)) \rceil)$. This bound is $\mathcal{O}(q)$ from the following facts:

$\sum_{t=1}^{\lceil 2 \log q \rceil} \frac{q}{2^t} \min(\lceil 1.1^t \rceil, \log q) t \log (t+1) \le q \sum_{t=1}^{\lceil 2 \log q \rceil} \frac{1.1^t + 1}{2^t} t \log (t+1) = \mathcal{O}(q)$,
and $\sum_{t=1}^{\lceil 2 \log q \rceil} q^{\frac{3}{4}} \min(\lceil 1.1^t \rceil, \log q) t \log (t+1) \le \sum_{t=1}^{\lceil 2 \log q \rceil} q^{\frac{3}{4}} (\log q) t \log (t+1) = \mathcal{O}(q)$.

From these bounds, the lemma follows. ■

We now get the query complexity of the algorithm.

Lemma 9 *Suppose that $\mathcal{C}$ is a set of n coins among which there are at most m counterfeits. Then, with probability $1 - \mathcal{O}\big(\frac{\log m}{m}\big)$,* Find-Counterfeits($\mathcal{C}$) *uses $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$ coin weighings.*

Proof By Lemma 6, with probability $1 - \mathcal{O}\big(\frac{\log m}{m}\big)$, (i) $\mathcal{C}_{\text{un}}$ contains at most $(\frac{2}{3})^{r-1} m$ counterfeits in the beginning of the r^{th} iteration of the **for** loop (inside Find-Counterfeits) for $r = 1, \ldots, \lceil \log \log m \rceil$, and (ii) it contains $\mathcal{O}\big(\frac{m}{\log m}\big)$ counterfeits right after the loop. Under this condition, by (i) and Lemma 8, the number of coin weighings over the iterations of the **for** loop is $\mathcal{O}\Big(\sum_{r=1}^{\lceil \log \log m \rceil} \frac{(\frac{2}{3})^{r-1} m \log n}{\log ((\frac{2}{3})^{r-1} m)}\Big) = \mathcal{O}\big(\frac{m \log n}{\log m}\big)$. Also by (ii), the total number of iterations of the **repeat** loop is $\mathcal{O}\big(\frac{m}{\log m}\big)$, and in each of the iterations, $\mathcal{O}(\log m + \log n)$ weighings are used (inside Find-One-Counterfeit). Thus, the number of coin weighings over the iterations of the **repeat** loop is $\mathcal{O}\big(\frac{m \log n}{\log m}\big)$. ■

4. Graph Finding Algorithm

Choi et al. (2011b) proposed an algorithmic framework for graph finding with additive queries based on a coin weighing algorithm, for positive real and bounded-integer weights. We modify the framework for cross-additive queries and combine it with the coin weighing algorithm in the previous section to give an algorithm for Theorem 1. The algorithm and analysis appear in the full version.

5. Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MOE) (No. 2011-0025650), and by NSF award CCF-1115703.

References

N. Alon and V. Asodi. Learning a hidden subgraph. *SIAM Journal on Discrete Mathematics*, 18(4):697–712, 2005.

N. Alon, R. Beigel, S. Kasif, S. Rudich, and B. Sudakov. Learning a hidden matching. *SIAM Journal on Computing*, 33(2):487–501, 2004.

D. Angluin and J. Chen. Learning a hidden graph using $\mathcal{O}(\log n)$ queries per edge. In *Proceedings of the 17th Annual Conference on Learning Theory (COLT 2004)*, pages 210–223, Banff, Canada, 2004.

D. Angluin and J. Chen. Learning a hidden hypergraph. *Journal of Machine Learning Research*, 7:2215–2236, 2006.

R. Beigel, N. Alon, M. S. Apaydin, L. Fortnow, and S. Kasif. An optimal procedure for gap closing in whole genome shotgun sequencing. In *Proceedings of the Fifth Annual International Conference on Computational Molecular Biology (RECOMB 2001)*, pages 22–30, 2001.

R. Berinde, A. C. Gilbert, P. Indyk, H. J. Karloff, and M. J. Strauss. Combining geometry and combinatorics: A unified approach to sparse signal recovery. *Allerton*, 2008.

N. H. Bshouty. Optimal algorithms for the coin weighing problem with a spring scale. In *Proceedings of the 22nd Annual Conference on Learning Theory (COLT 2009)*, Montreal, Canada, 2009.

N. H. Bshouty and H. Mazzawi. Toward a deterministic polynomial time algorithm with optimal additive query complexity. In *Proceedings of the 35th International Symposium on Mathematical Foundations of Computer Science (MFCS 2010)*, pages 221–232, Brno, Czech Republic, 2010a.

N. H. Bshouty and H. Mazzawi. Optimal query complexity for reconstructing hypergraphs. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010)*, pages 143–154, Nancy, France, 2010b.

N. H. Bshouty and H. Mazzawi. On parity check $(0, 1)$-matrix over $\mathbb{Z}_p$. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 1383–1394, San Francisco, USA, 2011a.

N. H. Bshouty and H. Mazzawi. Reconstructing weighted graphs with minimal query complexity. *Theoretical Computer Science*, 412(19):1782–1790, 2011b.

D. G. Cantor. Determining a set from the cardinalities of its intersections with other sets. *Canadian Journal of Mathematics*, 16:94–97, 1964.

D. G. Cantor and W. H. Mills. Determination of a subset from certain combinatorial properties. *Canadian Journal of Mathematics*, 18:42–48, 1966.

J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25(5):505–515, 1979a.

J. Capetanakis. Generalized TDMA: The multi-accessing tree protocol. *IEEE Transactions on Communications*, 27(10):1476–1484, 1979b.

S. S. Choi and J. H. Kim. Optimal query complexity bounds for finding graphs. In *Proceedings of the 40th ACM Symposium on Theory of Computing (STOC 2008)*, pages 749–758, Victoria, Canada, 2008.

S. S. Choi and J. H. Kim. Optimal query complexity bounds for finding graphs. *Artificial Intelligence*, 174(9–10):551–569, 2010.

S. S. Choi, K. Jung, and J. H. Kim. Almost tight upper bound for finding Fourier coefficients of k-bounded pseudo-Boolean functions. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT 2008)*, pages 123–134, Helsinki, Finland, 2008.

S. S. Choi, K. Jung, and B. R. Moon. Lower and upper bounds for linkage discovery. *IEEE Trans. on Evolutionary Computation*, 13(2):201–216, 2009.

S. S. Choi, K. Jung, and J. H. Kim. Almost tight upper bound for finding Fourier coefficients of k-bounded pseudo-Boolean functions. *Journal of Computer and System Sciences*, 77 (6):1039–1053, 2011a.

S. S. Choi, J. H. Kim, and J. Oh. Randomized polynomial time algorithms for finding weighted graphs with optimal additive query complexity. Manuscript, 2011b.

D. Du and F. K. Hwang. Combinatorial group testing and its application. In *V. 3 of Series on applied mathematics*, chapter 10. World Science, 1993.

P. Erdős and A. Rényi. On two problems of information theory. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 8:241–254, 1963.

A. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.

V. Grebinski. On the power of additive combinatorial search model. In *Proceedings of the 4th Annual International Conference on Computing and Combinatorics (COCOON 1998)*, pages 194–203, Taipei, Taiwan, 1998.

V. Grebinski and G. Kucherov. Reconstructing a Hamiltonian cycle by querying the graph: Application to DNA physical mapping. *Discrete Applied Mathematics*, 88:147–165, 1998.

V. Grebinski and G. Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28:104–124, 2000.

R. B. Heckendorn and A. H. Wright. Efficient linkage discovery by limited probing. *Evolutionary Computation*, 12(4):517–545, 2004.

J. J. Hein. An optimal algorithm to reconstruct trees from additive distance data. *Bulletin of Mathematical Biology*, 51(5):597–603, 1989.

P. Indyk and M. Ruzic. Near-optimal sparse recovery in the L_1 norm. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 199–207, 2008.

H. Kargupta and B. Park. Gene expression and fast construction of distributed evolutionary representation. *Evolutionary Computation*, 9(1):1–32, 2001.

J. H. Kim. Finding weighted graphs by combinatorial search. arXiv:1201.3793, 2012.

V. King, L. Zhang, and Y. Zhou. On the complexity of distance-based evolutionary tree reconstruction. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2003)*, pages 444–453, 2003.

B. Lindström. On a combinatory detection problem I. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 9:195–207, 1964.

B. Lindström. On a combinatorial problem in number theory. *Canadian Mathematical Bulletin*, 8(4):477–490, 1965.

B. Lindström. On Möbius functions and a problem in combinatorial number theory. *Canadian Mathematical Bulletin*, 14(4):513–516, 1971.

B. Lindström. Determining subsets by unramified experiments. In J. N. Srivastava, editor, *A Survey of Statistical Designs and Linear Models*, pages 407–418. North Holland, 1975.

J. L. Massey. Collision-resolution algorithms and random-access communications. In G. Longo, editor, *Multi-user communications systems, CISM Courses and Lecture Notes No. 265*, pages 73–137. Springer, Wien and New York, 1981.

H. Mazzawi. Optimally reconstructing weighted graphs using queries. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 608–615, Austin, USA, 2010.

C. McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, London Mathematical Society Lecture Note Series 141, pages 148–188. Cambridge University Press, 1989.

L. Moser. The second moment method in combinatorial analysis. In *Combinatorial Structures and Their Applications. Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications held at the University of Calgary. June 1969*, pages 283–384. Gordon and Breach, New York, 1970.

N. Pippenger. Bounds on the performance of protocols for a multiple-access broadcast channel. *IEEE Trans. on Information Theory*, 27(2):145–151, 1981.

L. Reyzin and N. Srivastava. Learning and verifying graphs using queries with a focus on edge counting. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory (ALT 2007)*, pages 285–297, Sendai, Japan, 2007a.

L. Reyzin and N. Srivastava. On the longest path algorithm for reconstructing trees from distance matrices. *Information Processing Letters*, 101(3):98–100, 2007b.

M. Ruszinkó and P. Vanroose. How an Erdös-Rényi-type search approach gives an explicit code construction of rate 1 for random access with multiplicity feedback. *IEEE Trans. on Information Theory*, 43(1):368–373, 1997.

S. Söderberg and H. S. Shapiro. A combinatory detection problem. *American Mathematical Monthly*, 70:1066–1070, 1963.

B. Tsybakov and V. Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Problemy Peredachi Informassi*, 14(4):259–280, 1978.

R. Uehara, K. Tsuchida, and I. Wegener. Identification of partial disjunction, parity, and threshold functions. *Theoretical Computer Science*, 210(1–2):131–147, 2000.

Appendix A. Proof of Lemma 5

To prove Lemma 5, we use the following lemma. Recall that for positive integers x and y, $L(x,y) = \sum_{j=0}^{x} \min(j,y)\binom{x}{j}$.

Lemma 10 *For positive integers x and y, $L(x,y) \geq \left(1 - \frac{1}{\sqrt{x}}\right) \min(\frac{x}{2}, y) 2^x$.*

Proof We consider only the case that x is even. The proof for odd x may be obtained in a similar way.

First, consider the case that $y \leq \frac{x}{2}$. We see that $L(x,y) = \sum_{j=0}^{y} j\binom{x}{j} + \sum_{j=y+1}^{x} y\binom{x}{j} = \sum_{j=0}^{x} y\binom{x}{j} - \sum_{j=0}^{y}(y-j)\binom{x}{j}$. Since $\sum_{j=0}^{y}(y-j)\binom{x}{j}/(y2^x)$ is increasing in y for $1 \leq y \leq \frac{x}{2}$, and $\sum_{j=0}^{\frac{x}{2}}(\frac{x}{2}-j)\binom{x}{j} = \frac{x}{4}\binom{x}{\frac{x}{2}}$, we have $\sum_{j=0}^{y}(y-j)\binom{x}{j}/(y2^x) \leq \sum_{j=0}^{\frac{x}{2}}(\frac{x}{2}-j)\binom{x}{j}/(\frac{x}{2}2^x) = \frac{x}{4}\binom{x}{\frac{x}{2}}/(\frac{x}{2}2^x) = \binom{x}{\frac{x}{2}}/2^{x+1}$ and so $L(x,y) = y2^x - \sum_{j=0}^{y}(y-j)\binom{x}{j} \geq y2^x - y2^x\binom{x}{\frac{x}{2}}/2^{x+1} \geq \left(1 - \frac{1}{\sqrt{x}}\right)y2^x$. Here, the last inequality follows from Stirling's formula.

Now, consider the case that $y > \frac{x}{2}$. For $y \geq x$, we see $L(x,y) = \sum_{j=0}^{x} j\binom{x}{j} = \frac{x}{2}2^x \geq \left(1-\frac{1}{\sqrt{x}}\right)\frac{x}{2}2^x$. For $\frac{x}{2} < y < x$, $L(x,y) = \sum_{j=0}^{x} j\binom{x}{j} - \sum_{j=y}^{x}(j-y)\binom{x}{j} = \frac{x}{2}2^x - \sum_{j=y}^{x}(j-y)\binom{x}{j}$, $\sum_{j=y}^{x}(j-y)\binom{x}{j} \leq \sum_{j=\frac{x}{2}}^{x}(j-\frac{x}{2})\binom{x}{j} = \frac{x}{4}\binom{x}{\frac{x}{2}}$, and so $L(x,y) \geq \frac{x}{2}2^x - \frac{x}{4}\binom{x}{\frac{x}{2}} \geq \left(1-\frac{1}{\sqrt{x}}\right)\frac{x}{2}2^x$. ■

Proof of Lemma 5 The main idea is to construct a $2^\nu \times L(\nu,y)$ search matrix M each column of which is the vector of the 2^ν values of a Boolean function defined on $\{-1,+1\}^\nu$. More specifically, for each $a \in \{0,1\}^\nu \setminus \{0^\nu\}$, let $\lambda_a := \min\left(|a|, y\right)$, where 0^ν is the vector consisting of ν zeros and $|a|$ is the number of 1's in a, and for $k = 1, \ldots, \lambda_a$, define $f_{a,k} : \{-1,+1\}^\nu \to \{0,1\}$ as follows:

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

for $x \in \{-1,+1\}^\nu$, where $j_1 < j_2 < \cdots < j_{|a|}$ are the positions of 1's in a. (The functions of this type were introduced in (Bshouty, 2009).) Then, we repeat the following for each $a \in \{0,1\}^\nu \setminus \{0^\nu\}$ in a non-increasing order of $|a|$. Choose the left-most λ_a columns in M among the ones not chosen so far, and put the value vectors of $f_{a,k}$'s from $k = 1$ to λ_a into the

λ_a columns, in the increasing order of column indices. We see that the number of columns in M is $\sum_{a\in\{0,1\}^\nu\setminus\{0^\nu\}}\lambda_a=\sum_{a\in\{0,1\}^\nu\setminus\{0^\nu\}}\min(|a|,y)=\sum_{j=0}^{\nu}\min(j,y)\binom{\nu}{j}=L(\nu,y)$.

Let $\mathbf{w}:=\big(w(A_1),0,w(A_2),0,\ldots,w(A_q),0,0,\ldots,0\big)^\top$ which is a vector of size $L(\nu,y)$ with $L(\nu,y)-q$ zeros, and for each $a\in\{0,1\}^\nu\setminus\{0^\nu\}$ and $k=1,\ldots,\lambda_a$, let $w_{a,k}$ be the i^{th} coordinate of $\mathbf{w}$ such that the i^{th} column in M corresponds to $f_{a,k}$. (If such i is odd and at most $2q-1$, $w_{a,k}=w(A_{(i+1)/2})$ and otherwise, $w_{a,k}=0$.) Then, $M\mathbf{w}$ is the 2^ν-vector of the values of a pseudo-Boolean function $f:\{-1,+1\}^\nu\to\{0,1,2,\ldots\}$ defined as

$$f(x)=\sum_{a\in\{0,1\}^\nu\setminus\{0^\nu\}}\sum_{k=1}^{\lambda_a}w_{a,k}f_{a,k}(x)$$

for $x\in\{-1,+1\}^\nu$.

To get the desired equation for each $w(A_i)$ (or equivalently, for the corresponding $w_{a,k}$), we use a Fourier transform of f with regard to the basis functions $\chi_a:\{-1,+1\}^\nu\to\{-1,+1\}$ for $a\in\{0,1\}^\nu$ defined as $\chi_a(x)=\prod_{i:a_i=1}x_i$ for $x\in\{-1,+1\}^\nu$. More specifically, for any $a\in\{0,1\}^\nu$, the Fourier coefficient $\widehat{f}(a)$ of χ_a in f is

$$\begin{aligned}\widehat{f}(a)&=\sum_{b\in\{0,1\}^\nu\setminus\{0^\nu\}}\sum_{k=1}^{\lambda_b}w_{b,k}\widehat{f}_{b,k}(a)\\&=\sum_{b\supsetneq a}\sum_{k=1}^{\lambda_b}w_{b,k}\widehat{f}_{b,k}(a)+\sum_{k=1}^{\lambda_a}w_{a,k}\widehat{f}_{a,k}(a)\\&=\sum_{b\supsetneq a}\sum_{k=1}^{\lambda_b}w_{b,k}\widehat{f}_{b,k}(a)+\sum_{k=1}^{\lambda_a}\frac{w_{a,k}}{2^k}.\end{aligned}$$

Here, a vector in $\{0,1\}^\nu$ is regarded as the set of the positions of 1's in the vector, for set comparison. The second equality follows from the fact that for any $b\in\{0,1\}^\nu$ with $b\not\supset a$, $\widehat{f}_{b,k}(a)=0$ for $k=1,\ldots,\lambda_b$. The third equality follows from the fact that for $k=1,\ldots,\lambda_a$, $\widehat{f}_{a,k}(a)=2^{-k}$. Thus for each $w(A_i)$, if $w(A_i)$ corresponds to $w_{a,k}$, we have the equation

$$w(A_i)=w_{a,k}=2^k\Big(\widehat{f}(a)-\sum_{b\supsetneq a}\sum_{j=1}^{\lambda_b}w_{b,j}\widehat{f}_{b,j}(a)-\sum_{j=1}^{k-1}\frac{w_{a,j}}{2^j}\Big)-\sum_{j=1}^{\lambda_a-k}\frac{w_{a,k+j}}{2^j}.$$

By the way of mapping the Boolean functions to the columns in M mentioned above, we see that $w_{b,j}$ for $b\supsetneq a$ and $j=1,\ldots,\lambda_b$ and $w_{a,j}$ for $j<k$ correspond to some $w(A_{i'})$'s with $i'<i$ or 0, and for $j=1,\ldots,\lambda_a-k$, $w_{a,k+j}$ corresponds to 0 for odd j and to $w(A_{i+j/2})$ or 0 for even j. Since $\lambda_a-k<\lambda_a\le\min(\nu,y)$, we have a desired equation for $w(A_i)$. In particular, the values $\widehat{f}(a)$ for $a\in\{0,1\}^\nu$ may be all found in polynomial in q (by the fast Fourier transform of f) with 2^ν non-adaptive coin weighings (for computing $M\mathbf{w}$). The values $\widehat{f}_{b,j}(a)$ for $a,b\in\{0,1\}^\nu$ and $j=1,\ldots,\lambda_b$ may be all found in polynomial in q as well (by the definition of $f_{b,j}$), without any weighing. Thus, we may use 2^ν non-adaptive coin weighings to get desired equations for $w(A_i)$'s in polynomial time in q.

Finally to bound 2^ν, we assume $\nu\ge2$ and see that $L(\nu-1,y)<2q$ by the choice of ν for $L(\nu,y)$. Also by Lemma 10, we see $L(\nu-1,y)\ge\big(1-\frac{1}{\sqrt{\nu-1}}\big)\min(\frac{\nu-1}{2},y)2^{\nu-1}$

and so $\left(1-\frac{1}{\sqrt{\nu-1}}\right)\min(\frac{\nu-1}{2},y)2^{\nu-1} < 2q$. This means that $\left(1-\frac{1}{\sqrt{\nu-1}}\right)\frac{\nu-1}{2}2^{\nu-1} < 2q$ or $\left(1-\frac{1}{\sqrt{\nu-1}}\right)2^{\nu-1} < \frac{2q}{y}$. From this and by using a calculus, we may see that there is a constant $\alpha > 0$ such that $2^\nu \le \left(1+\frac{\alpha}{\sqrt{\log 2q}}\right)\max\left(\frac{4q}{y}, \frac{8q}{\log 2q}\right)$ for sufficiently large q. ■

Appendix B. Proof of Lemma 6

To prove Lemma 6, we need a few lemmas. We start with a lemma on the distribution of coins in Find-Portion-of-Counterfeits. (A similar analysis was given in (Choi et al., 2011b).)

Lemma 11 *Suppose that $\mathcal{C}$ is a set of coins among which there are at most q counterfeits. Then, the following holds inside the call* Find-Portion-of-Counterfeits($\mathcal{C}$,q):

(a) *With probability $1-e^{-\Omega\left(q^{1/2}\right)}$, for all $t=0,\ldots,\lceil 2\log q\rceil-1$,*

$$(\textit{the number of sets in } \mathcal{S}_t \textit{ with two or more counterfeits}) \le \frac{q}{2^{t+1}} + q^{3/4}.$$

(b) *With probability $1-\mathcal{O}\left(\frac{1}{q}\right)$, for all $t=\lceil 2\log q\rceil,\ldots,\lceil 2\log q\rceil+\lceil 2\log n\rceil-1$,*

$$(\textit{the number of sets in } \mathcal{S}_t \textit{ with two or more counterfeits}) = 0.$$

(c) *With probability $1-e^{-\Omega(q)}$, the number of counterfeits each of which belongs to a set in $\mathcal{S}_0$ with two or more counterfeits is at most $\frac{2}{3}q$.*

To prove (a) and (c), we use a martingale inequality from (McDiarmid, 1989) as follows.

Lemma 12 (McDiarmid, 1989) *Let $X=(X_1,\ldots,X_\ell)$ be a family of independent random variables with X_i taking values in a finite set $\mathcal{A}_i$ for each i. Suppose that the real-valued function f defined on $\prod_i \mathcal{A}_i$ satisfies*

$$|f(\mathbf{x})-f(\mathbf{x}')| \le c_i$$

whenever the vectors $\mathbf{x}$ and $\mathbf{x}'$ differ only in the i-th coordinate. Then for any $\lambda \ge 0$,

$$\Pr\left[|f(X)-\mathrm{E}[f(X)]| \ge \lambda\right] \le 2e^{-2\lambda^2/\sum_i c_i^2}.$$

Proof of Lemma 11 For (a), first notice that each S in $\mathcal{S}_t$ is a random set obtained by putting each coin in $\mathcal{C}$ into S with probability $\frac{1}{q2^t}$ independently of other coins. More precisely, letting q_0 be the number of counterfeits in $\mathcal{C}$, the number of sets in $\mathcal{S}_t$ with two or more counterfeits is the same as the number of sets with two or more counterfeits among $q2^t$ random sets into one of which each of q_0 counterfeits is put uniformly at random and independently of other counterfeits. Thus, if we denote the number of sets by F,

$$\mathrm{E}[F] \le q2^t \times \binom{q_0}{2}\left(\frac{1}{q2^t}\right)^2 \le \frac{q}{2^{t+1}}.$$

Label the counterfeits and the random sets as 1 through q_0 and as 1 through $q2^t$, respectively. Let $X_1,\ldots,X_{q_0}$ be the random variables such that $X_i=k$ if the counterfeit i is put into

the set k. Then, F is a function depending only on the independent random variables X_i's, and the value of F changes by at most 1 when only one of X_i's changes. Thus, Lemma 12 with $\lambda = q^{3/4}$ and $c_i = 1$ for $i = 1, \ldots, q_0$ gives

$$\Pr\left[F \geq \frac{q}{2^{t+1}} + q^{3/4}\right] \leq 2e^{-2q^{\frac{3}{2}}/q_0} \leq 2e^{-2q^{\frac{1}{2}}},$$

and (a) follows by union bound.

For (b), we should notice that if there is no set with two or more counterfeits in $\mathcal{S}_{\lceil 2\log q\rceil}$, then the same is true in $\mathcal{S}_t$ for any $t > \lceil 2\log q\rceil$. Thus, (b) follows from the following:

$$\Pr[\,\exists \text{ a set in } \mathcal{S}_{\lceil 2\log q\rceil} \text{ with two or more counterfeits}\,] \leq q2^{\lceil 2\log q\rceil} \times \binom{q_0}{2}\left(\frac{1}{q2^{\lceil 2\log q\rceil}}\right)^2 \leq \frac{1}{2q}.$$

For (c), we may assume that q_0 is not zero. Then, for each counterfeit c in $\mathcal{C}$,

$$\Pr[\,c \text{ is the only counterfeit in } S \text{ for some } S \in \mathcal{S}_0\,] = (1-1/q)^{q_0-1} \geq (1-1/q)^{q-1} \geq 1/e,$$

and so the expected number of counterfeits each of which belongs to a set in $\mathcal{S}_0$ with two or more counterfeits is at most $(1-1/e)q_0 \leq (1-1/e)q$. Thus, we have that the number is at most $2q/3$ with probability $1 - e^{-\Omega(q)}$ by using Lemma 12 (in a similar way as in the proof of (a)). ■

The following gives a large-deviation inequality for a sum of independent positive-integer-valued random variables of which probability masses are bounded above by a geometrically decreasing sequence. It will be used in the next lemma to prove the error bound for FIND-PORTION-OF-COUNTERFEITS.

Lemma 13 *Let $X_1, \ldots, X_\ell$ be positive-integer-valued independent random variables such that for some $0 < \alpha < 1$ and $\beta > 0$, each X_i satisfies $\Pr[X_i = k] \leq \beta\alpha^k$ for $k = 1, 2, \ldots$, and let $X := \sum_{i=1}^{\ell} X_i$. Then for any $\delta > 0$,*

$$\Pr\left[\, X \geq (1+\delta)\frac{\ell}{1-\alpha}\,\right] \leq \left(\frac{\beta\alpha}{1-\alpha}\cdot\frac{1+\delta}{e^\delta}\right)^\ell.$$

Proof Let $Y_1, \ldots, Y_\ell$ be independent geometric random variables with parameter $1-\alpha$, and let $Y := \sum_{i=1}^{\ell}$. Then, for each $i = 1, \ldots, \ell$ and $k = 1, 2, \ldots$, $\Pr[X_i = k] \leq \frac{\beta\alpha}{1-\alpha}(1-\alpha)\alpha^{k-1} = \frac{\beta\alpha}{1-\alpha}\Pr[Y_i = k]$, and so

$$\Pr\left[\, X \geq (1+\delta)\frac{\ell}{1-\alpha}\,\right] \leq \left(\frac{\beta\alpha}{1-\alpha}\right)^\ell \Pr\left[\, Y \geq (1+\delta)\frac{\ell}{1-\alpha}\,\right].$$

The lemma follows from the fact that

$$\Pr\left[\, Y \geq (1+\delta)\frac{\ell}{1-\alpha}\,\right] \leq \left(\frac{1+\delta}{e^\delta}\right)^\ell.$$

This inequality is proven by a standard technique using the moment generating function of Y. In more detail, for any t with $0 < t < \ln\frac{1}{\alpha}$,

$$\Pr\left[\, Y \geq (1+\delta)\frac{\ell}{1-\alpha}\,\right] \leq \frac{\mathrm{E}[e^{tY}]}{e^{\frac{t(1+\delta)\ell}{1-\alpha}}} = \left(\frac{(1-\alpha)e^t}{(1-\alpha e^t)e^{\frac{t(1+\delta)}{1-\alpha}}}\right)^\ell.$$

Here, the inequality follows by Markov inequality for $t > 0$, and the equality follows from the fact that $\mathrm{E}[e^{tY}] = \prod_{i=1}^{\ell} \mathrm{E}[e^{tY_i}] = \left(\frac{(1-\alpha)e^t}{1-\alpha e^t}\right)^{\ell}$ for $t < \ln \frac{1}{\alpha}$. By choosing $t = \ln \frac{\delta+\alpha}{\alpha(1+\delta)}$, we have

$$\frac{(1-\alpha)e^t}{(1-\alpha e^t)e^{\frac{t(1+\delta)}{1-\alpha}}} = (1+\delta)\left(1 - \frac{\delta(1-\alpha)}{\delta+\alpha}\right)^{\frac{\delta+\alpha}{1-\alpha}} \leq (1+\delta)e^{-\delta}$$

and get the inequality in the above, as desired. ■

In the following, we bound the error probability for Find-Portion-of-Counterfeits by considering the absorption time of a Markov chain.

Lemma 14 *Suppose that $\mathcal{C}$ is a set of coins among which there are at most q counterfeits. Also, suppose that inside the call* Find-Portion-of-Counterfeits($\mathcal{C}$,q)*, for fixed t between 1 and $\lceil 2\log q \rceil$ and for each $S_i \in \mathcal{S}_{t-1}$, $u(S_i)$ is equal to the weight of S_i in the beginning of the* **for** *loop of lines 3–33. Then with probability $1 - e^{-\Omega(q^{1/2})}$, $u(S_{ij})$ is equal to the weight of S_{ij} for all i, j in the end of the loop.*

Proof We may consider the process of computing (and updating) $u(S_{ij})$'s in lines 9–28 as a series of transitions on an MC defined as follows: Each state in the chain corresponds to a situation right after an interval is selected by the algorithm (for initial setting or through a roll-back or a guess/check-and-fix operation). It is specified by a pair of an interval I and the weights of $S_{i1}, i \in \mathrm{pre}(I)$ being stored when I is selected.

The **repeat** loop starts with the state $(I_1, \textsc{Null})$. (Notice that $\mathrm{pre}(I_1)$ is empty as $\min(I_1) = 1$.) Suppose that I_h is now selected, and the weight of S_{i1} for $i \in \mathrm{pre}(I_h)$ being currently stored is $\dot{u}(S_{i1})$. Then, in a pass of the **repeat** loop of lines 10–28, the algorithm moves from the state $\left(I_h, \left(\dot{u}(S_{i1})\right)_{i\in\mathrm{pre}(I_h)}\right)$ to the state $\left(I_{h-1}, \left(\dot{u}(S_{i1})\right)_{i\in\mathrm{pre}(I_{h-1})}\right)$ if a roll-back occurs, and moves to the state $\left(I_{h+1}, \left(\dot{u}(S_{i1})\right)_{i\in\mathrm{pre}(I_{h+1})}\right)$ otherwise, where $\dot{u}(S_{i1})$ for $i \in I_h$ is the weight of S_{i1} computed in this pass. Notice that the only absorbing state in this MC is $\left(\emptyset, \left(\dot{u}(S_{i1})\right)_{i\in[1,q_t]}\right)$ with $\dot{u}(S_{i1}) = w(S_{i1})$ for all i. This corresponds to the situation in which all incorrect guesses, if existed, have been fixed.

Consider the *royal road*, i.e., the path from the starting state to the absorbing one which is obtained by the series of transitions occurring in the case that Check-All-Zeros works correctly every time it is called inside the **repeat** loop. Let ℓ_R be the distance between the starting and absorbing states on the royal road. Let s_0 be the starting state, and for $k = 1, \ldots, \ell_R$, let s_k be the k^{th} state from the starting state on the royal road. For $k = 1, \ldots, \ell_R$, let X_k be the number of transitions made until s_k is reached from the first visit of s_{k-1}. Letting $X := \sum_{k=1}^{\ell_R} X_k$, we show that for a constant $C > 0$, say $C = 9$,

$$\Pr\left[X \geq C\left(\frac{q}{2^t} + q^{\frac{3}{4}}\right)\min(\lceil 1.1^t \rceil, \log q)\right] = e^{-\Omega\left(q^{\frac{1}{2}}\right)}$$

from which the lemma follows. (Here, we do not optimize C.)

The distance ℓ_R is bounded above by the number of leading incorrect guesses (for transitions with fixing incorrect guesses) plus $\lceil \frac{q}{2^t} \rceil$ (for transitions without fixing guesses). The number of leading incorrect guesses is at most $\left(\frac{q}{2^t} + q^{\frac{3}{4}}\right)\min(\lceil 1.1^t \rceil, \log q)$ with probability

$1 - e^{-\Omega(q^{1/2})}$ by Lemma 11 (a), Lemma 5, and the arguments at the bottom of Section 3.1. Thus, with probability $1 - e^{-\Omega(q^{1/2})}$, we have $\ell_R \le \frac{3}{2}\left(\frac{q}{2^t} + q^{\frac{3}{4}}\right) \min(\lceil 1.1^t \rceil, \log q)$.

On the other hand, notice that for $k = 1, \ldots, \ell_R$, once s_k is visited, s_{k_0} is not visited any more for any $k_0 < k$. From the setting of K values for CHECK-ALL-ZEROS in the **repeat** loop and in FIND-FIRST-INCORRECT, we use the reflection principle for random walks to see that for a positive integer z and $\varepsilon = \frac{1}{16}$,

$$\Pr[\, X_k = z \,] \begin{cases} \le \frac{1}{z}\binom{z}{\frac{z+1}{2}}\varepsilon^{\frac{z-1}{2}} & \text{if } z \text{ is odd,} \\ = 0 & \text{otherwise.} \end{cases}$$

In particular, for odd z, we have $\frac{1}{z}\binom{z}{\frac{z+1}{2}}\varepsilon^{\frac{z-1}{2}} = \frac{2}{z+1}\binom{z-1}{\frac{z-1}{2}}\varepsilon^{\frac{z-1}{2}} \le 2^{z-1}\varepsilon^{\frac{z-1}{2}} = 2\left(\frac{1}{2}\right)^z$. Thus, under the condition that $\ell_R \le \frac{3}{2}\left(\frac{q}{2^t} + q^{\frac{3}{4}}\right) \min(\lceil 1.1^t \rceil, \log q)$, we use Lemma 13 with $\alpha = \frac{1}{2}$, $\beta = 2$, and $2(1+\delta)\ell_R = C\left(\frac{q}{2^t} + q^{\frac{3}{4}}\right) \min(\lceil 1.1^t \rceil, \log q)$ to see that the probability of $X \ge C\left(\frac{q}{2^t} + q^{\frac{3}{4}}\right) \min(\lceil 1.1^t \rceil, \log q)$ is $e^{-\Omega(q^{3/4})}$, e.g., for $C = 9$. (We omit the detail.) Thus, the desired inequality follows by considering the probability of the condition as well. ■

Now we prove Lemma 6.

Proof of Lemma 6 Consider inside the call FIND-PORTION-OF-COUNTERFEITS($\mathcal{C}$,q). First, the following holds with probability $1 - \mathcal{O}\left(\frac{1}{q}\right) - e^{-\Omega(q^{1/2})} - e^{-\Omega(q)}$: (i) each of the counterfeits in $\mathcal{C}$ except at most $\frac{2}{3}q$ ones is the only counterfeit belonging to a set in $\mathcal{S}_0$, and (ii) for all $t = 1, \ldots, \lceil 2\log q \rceil + \lceil \log n \rceil$, $\mathcal{S}_t$ contains all and only the sets S such that S is a set of non-zero weight obtained by the division of a set in $\mathcal{S}_{t-1}$, and for all $S \in \mathcal{S}_t$, $u(S)$ is the same as the weight of S. We may get this by using Lemma 11 (c), Lemma 14 (inductively on $t = 1, \ldots, \lceil 2\log q \rceil$), and Lemma 11 (b) (on $t = \lceil 2\log q \rceil + 1, \ldots, \lceil 2\log q \rceil + \lceil \log n \rceil$), the latter of which implies that no set in $\mathcal{S}_t$ is split for all $t = \lceil 2\log q \rceil, \ldots, \lceil 2\log q \rceil + \lceil \log n \rceil - 1$. This means that with probability $1 - \mathcal{O}\left(\frac{1}{q}\right)$, each $c \in \mathcal{C}_{\mathrm{fd}}$ is a counterfeit, $u(c)$ is the same as the weight of c, and the size of $\mathcal{C} \setminus \mathcal{C}_{\mathrm{fd}}$ is at most $\frac{2}{3}q$. This completes the proof. ■