Theoretical Computer Science 456 (2012) 51–64

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Quantum counterfeit coin problems☆

Kazuo Iwama [a], Harumichi Nishimura [b,*], Rudy Raymond [c], Junichi Teruyama [a]

[a] School of Informatics, Kyoto University, Yoshida-Honmachi, Kyoto 606-8501, Japan

[b] Graduate School of Information Science, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

[c] IBM Research – Tokyo, 1623-14 Shimotsuruma, Yamato 242-8502, Japan

ARTICLE INFO

Article history:
Received 31 December 2010
Received in revised form 10 May 2012
Accepted 26 May 2012
Communicated by C.S. Calude

Keywords:
Counterfeit coin problems
Quantum computing
Query complexity

ABSTRACT

The counterfeit coin problem requires us to find all false coins from a given bunch of coins using a balance scale. We assume that the balance scale gives us only "balanced" or "tilted" information and that we know the number k of false coins in advance. The balance scale can be modeled by a certain type of oracle and its query complexity is a measure for the cost of weighing algorithms (the number of weighings). In this paper, we study the quantum query complexity for this problem. Let $Q(k, N)$ be the quantum query complexity of finding all k false coins from the N given coins. We show that for any k and N such that $k < N/2$, $Q(k, N) = O(k^{1/4})$, contrasting with the classical query complexity, $\Omega(k \log(N/k))$, that depends on N. So our quantum algorithm achieves a *quartic* speed-up for this problem. We do not have a matching lower bound, but we show some evidence that the upper bound is tight: any algorithm, including our algorithm, that satisfies certain properties needs $\Omega(k^{1/4})$ queries.

1. Introduction

Exponential speed-ups by quantum algorithms have been highly celebrated, but their specific examples are not too many. In contrast, almost every unstructured search problem can be sped up simply by using amplitude amplification [6,7,9], providing a huge number of combinatorial problems for which quantum algorithms are *quadratically* faster than classical ones. Interestingly there are few examples in between. (For instance, [8] provides a cubic speed-up while their classical lower bound is not known.) The reason is probably that the amplitude amplification is too general to combine with other methods appropriately. In fact we know few such cases including the one by [16] where they improved a simple Grover search algorithm for triangle finding by using clever combinatorial ideas (but unfortunately still less than quadratically compared to the best classical algorithm). This paper achieves a *quartic* speed-up for a well-known combinatorial problem.

The counterfeit coin problem is a mathematical puzzle whose origin dates back to 1945; in the American Mathematical Monthly, 52, p. 46, E. Schell posed the following question which is probably one of the oldest questions about the complexity of algorithms: "You have eight similar coins and a beam balance. At most one coin is counterfeit and hence underweight. How can you detect whether there is an underweight coin, and if so, which one, using the balance only twice?" The puzzle immediately fascinated many people and since then there have been several different versions and extensions in the literature (see e.g., [10,11,15,17]).

☆ An extended abstract of this article was presented in Proceedings of 21st ISAAC, Lecture Notes in Computer Science, vol. 6506, pp. 85–96, 2010.

* Corresponding author.
E-mail addresses: iwama@kuis.kyoto-u.ac.jp (K. Iwama), hnishimura@is.nagoya-u.ac.jp (H. Nishimura), raymond@jp.ibm.com (R. Raymond), teruyama@kuis.kyoto-u.ac.jp (J. Teruyama).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.05.039

This paper considers the quantum version of this problem, which, a bit surprisingly, has not appeared in the literature. To make our model simple, we assume that we cannot obtain information on which side is heavier when the balance scale is tilted. So, the balance scale gives us only binary information, *balanced* (i.e., two sets of coins on the two pans are equal in weight) or *tilted* (different in weight). Our goal is to detect the false coin with a minimum number of weighings. The problem is naturally extended to the case that there are two or more ($=k$ that is known in advance) false coins with equal weight. For the simplest case that $k = 1$, the following easy (classic) algorithm exists: we put (approximately) $N/4$ coins on each pan. If the balance scale is tilted, then we know the false coin is in those $N/2$ coins and if it is balanced, then the false one should be in the remaining $N/2$ ones. Also, it is easy to see that two weighings are enough for $N = 4$. Thus $\lceil \log N \rceil$ weighings are enough for $k = 1$ and this is also an information theoretic lower bound. (The original version of the problem assumes ternary outputs from the balance scale: left-heavy, right-heavy and balanced, and that the false coin is always underweight. As one can see easily, however, the same idea allows us to obtain the tight $\lceil \log_3 N \rceil$ upper bound.)

Our model of a balance scale is a so-called *oracle*. *A balance oracle* or simply *a B*-oracle is an N-bit register, which includes (originally unknown) N bits, $x_1x_2 \cdots x_N \in \{0, 1\}^N$. In order to retrieve these values, we can make *a query* with *a query string* $q_1q_2 \cdots q_N \in \{0, 1, -1\}^N$ including the same number of 1's and (-1)'s. Then the oracle returns a one-bit answer χ defined as:

$$\chi = 0 \quad \text{if } x_1q_1 + \cdots + x_Nq_N = 0 \quad \text{and} \quad \chi = 1 \text{ otherwise.}$$

Consider $x_1, \ldots, x_N$ as N coins where 0 means a fair coin and 1 a false one. Then, $q_i = 1$ means we place coin x_i on the left pan and $q_i = -1$ on the right pan. Since we must have the same number of 1's and (-1)'s, the answer χ correctly simulates the balance scale, i.e., $\chi = 0$ means it is balanced and $\chi = 1$ tilted. The number of weighings needed to retrieve x_1 through x_N (or to identify all the false coins) is called *query complexity*.

The main purpose of this paper is to obtain *quantum* query complexity for the counterfeit coin problems. Observe that if we know in advance that an even-cardinality set X includes *at most one* false coin, then by using the balance scale for any equal-size partition of X we can get the parity of X, i.e., the parity of the number (zero or one, now) of false coins in X. This means that for strings including at most one 1, the B-oracle is equivalent to the so-called *Inner Product oracle* (or IP oracle) [5]. Therefore, by Bernstein–Vazirani algorithm [5], we need only one weighing to detect the false coin. (Note that this observation was essentially done by Terhal and Smolin [22].) This already allows us to design the following quantum algorithm for general k: recall that we know k in advance. So, if we sample N/k coins at random, then they include exactly one false coin with high probability and we can find it using the B-oracle just once as mentioned above. Thus, by using the standard amplitude amplification [7] (together with careful consideration for the answer-confirmation procedure), we need $O(k)$ weighings to find all k false coins. For a small k, this is already much better than $\Omega(k \log(N/k))$ that is an information theoretic lower bound for the classical case.

Our contribution. This paper shows that this complexity can be furthermore improved quartically, namely, our new algorithm needs $O(k^{1/4})$ weighings. Note that the above idea, the one exploiting Bernstein–Vazirani's, already breaks down for $k = 2$, since the balance scale tilts even if the pans hold two (even) false coins if they both go to a same pan. Moreover, if k grows, say as large as linear in N, the balance scale will be tilted almost always for randomly selected equal partitions. Nevertheless, Bernstein–Vazirani's is useful since it essentially reduces our problem (identifying false coins) to the problem of deciding the parity of the number of the false coins that turns out to be an easier task for B-oracles. By this we can get a single quadratic speed-up and another quadratic speed-up by amplitude amplification.

We conjecture that this bound is tight, but unfortunately, we cannot prove it at this moment. The main difficulty is that we have a lot of freedom on "the size of the pans" (= the number of coins placed on the two pans of the balance scale), which makes it hard to design a single weight scheme of the adversary method [1]. However, we do have a proof claiming that it would be hard to do better unless we can remove the two fundamental properties of our algorithm. These properties are (i) the big-pan property and (ii) the random-partition property. We have considered several possibilities for removing them, but were not successful for even one of them.

Related work. Query complexities have been studied almost always for the standard *index oracle*, which accepts an index i and returns the value of x_i. Other than this oracle, we know few ones including the IP oracle [5] mentioned before and the even more powerful one that returns the number (not the parity) of 1's in the string [22]. Also, [22] presented a single-query quantum algorithm for the binary search problem under the IP oracle, which is essentially based on the same idea as the $k = 1$ case of our problem mentioned above.

The quantum adversary method, which is used for B-oracles in this paper, was first introduced by Ambainis [1] for the standard oracle. Many variants have followed including weighted adversary methods [2,23], spectral adversary method [4], Kolmogorov complexity method [13], all of which were shown to be equivalent [21]. After Høyer et al. [12] introduced a stronger quantum adversary method called the negative adversary method, Reichardt [19,20] showed that this method characterizes the quantum query complexity up to constant factors for any Boolean function. See [3,14] for further recent developments.

Models. A *B-oracle* is a binary string $x = x_1 \cdots x_N$ where $x_i = 1$ (resp. $x_i = 0$) means that the i-th coin is false (resp. fair). For instance, the string 0001 for $N = 4$ means that the fourth coin is a unique false coin. A query to the oracle is given as a string $q = q_1 \cdots q_N \in \{0, 1, -1\}^N$ that must be in the set $\bigcup_{l=1}^{\lfloor N/2 \rfloor} Q_l$ where Q_l is the set of strings q such that q has exactly l 1's and l (-1)'s. Here, 1 (or -1, resp.) in the i-th component means that we place the i-th coin on the left pan (on the right pan, resp.) and 0 means that the i-th coin is not placed on either pan. The answer from the oracle is represented by a binary value

$\chi_x(q)$ where $\chi_x(q) = 0$ means the balance scale is balanced, that is, $q_1x_1 + \cdots + q_Nx_N = 0$ and $\chi_x(q) = 1$ means it is tilted, that is, $q_1x_1 + \cdots + q_Nx_N \neq 0$. In quantum computation, the B-oracle is viewed as a unitary transformation $O_{B,x}$. Namely, $O_{B,x}$ transforms $|q\rangle$ to $(-1)^{\chi_x(q)}|q\rangle$. Throughout this paper, we assume that the Hamming weight k of the B-oracle x, denoted as $\mathrm{wt}(x) = k$, is less than $N/2$ since our B-oracle model is unable to distinguish x from $\bar{x}$ (the bit string obtained by flipping all bits of x).

2. Upper bounds

Here is our main result in this paper:

Theorem 1. *The quantum query complexity for finding the k false coins among N coins is $O(k^{1/4})$.*

Notice that our algorithm is *exact*, i.e., its output must be correct with probability one to compare our result with the classical case (which has been often studied in the exact setting). Since we use exact amplitude amplification [7] to make our algorithm exact, the assumption that k is known is necessary. But it should be noted that our bounded-error algorithm described in this section works even for unknown k. Also, we note that our algorithm can be easily adapted so that it works when the output of the balance scale is ternary (while we assume it is binary for simplicity).

Before the proof, we first describe our basic approach, a simulation of the IP oracle by the B-oracle. Recall that the IP oracle (Inner Product oracle) [5] transforms a prequery state $|\widetilde{q}\rangle_{\mathsf{R}}$ to $(-1)^{\widetilde{q}\cdot x}|\widetilde{q}\rangle_{\mathsf{R}}$, where $\widetilde{q} \in \{0,1\}^N$ in register R is a query string and $x \in \{0,1\}^N$ is an oracle. Then the Bernstein–Vazirani algorithm (the Hadamard transform) retrieves the string x and we know the k false coins in the case of our problem. Observe that the IP oracle flips the phase of $|\widetilde{q}\rangle_{\mathsf{R}}$ if and only if $\widetilde{q}\cdot x$ is odd, in other words, if and only if the set $\{i \mid \widetilde{q}_i = 1\}$ includes an odd number of indexes such that $x_i = 1$. Note that an index i such that $x_i = 1$ is a false coin in our case. If $k = 1$, then $\{i \mid \widetilde{q}_i = 1\}$ includes at most one false coin. Hence we can simply replace the IP oracle with the query string $\widetilde{q}$ by the B-oracle with a query string q such that an arbitrary one half (the first one half, for instance) of the 1's in $\widetilde{q}$ are changed to (-1)'s, which means that the one half of the coins in $\{i \mid \widetilde{q}_i = 1\}$ (i.e., the set of coins placed on the pans) go to the left pan and the remaining one half to the right pan. (As shown in a moment, we can assume without loss of generality that $\mathrm{wt}(\widetilde{q})$ is even.)

Now we consider the general ($k \geq 1$) case. If $\{i \mid \widetilde{q}_i = 1\}$ includes an odd number of false coins, then the balance scale is tilted for any such q mentioned above; this is desirable for us. If $\{i \mid \widetilde{q}_i = 1\}$ includes an even number of false coins, we wish the balance scale to be balanced. In order for this to happen, however, we must divide the (unknown) false coins in $\{i \mid \widetilde{q}_i = 1\}$ into the two pans evenly, for which there are no obvious ways other than using randomization. Our idea is to introduce the second register R' as follows: on R', we prepare, conditionally on register R being in state $|\widetilde{q}\rangle$, a superposition of all possible states $q_1(\widetilde{q}), q_2(\widetilde{q}), \ldots, q_h(\widetilde{q})$, obtained by flipping one half of 1's in $\widetilde{q}$ into (-1)'s. By using this superposition as a query to the B-oracle, we can achieve a success (being able to detect if the balance scale is balanced) probability of $1/\sqrt{m}$, where m is the number of false coins in $\{i \mid \widetilde{q}_i = 1\}$. In order to increase this probability, we can use copies of register R' or, more efficiently, quantum amplitude amplification [7].

As suggested before, we begin with the restriction of the IP oracle without losing its power. A *parity-restricted query* in the following lemma means a superposition of query strings whose Hamming weights are even.

Lemma 1. *Let $S_{<N/2} := \{x \in \{0,1\}^N \mid \mathrm{wt}(x) < N/2\}$. Then there is a quantum algorithm to identify an oracle in $S_{<N/2}$ by a single parity-restricted query to the IP oracle.*

Proof. For a given oracle $x \in S_{<N/2}$, define

$$|\psi_x\rangle = \frac{1}{\sqrt{2^{N-1}}} \sum_{\widetilde{q}\in Q_{\mathrm{ew}}} (-1)^{\widetilde{q}\cdot x}|\widetilde{q}\rangle,$$

where $Q_{\mathrm{ew}} = \{\widetilde{q} \in \{0,1\}^N \mid \mathrm{wt}(\widetilde{q}) = 0 \bmod 2\}$. Then the state after applying the Hadamard transform $H^{\otimes N}$ on $|\psi_x\rangle$ (where H is the Hadamard gate [18]) can be rewritten as follows:

$$\begin{aligned}
H^{\otimes N}|\psi_x\rangle &= \frac{1}{\sqrt{2^{N-1}}} \sum_{\widetilde{q}\in Q_{\mathrm{ew}}} (-1)^{\widetilde{q}\cdot x} H^{\otimes N}|\widetilde{q}\rangle \\
&= \frac{1}{2^{N-1}\sqrt{2}} \sum_{\widetilde{q}\in Q_{\mathrm{ew}}} \sum_{z\in\{0,1\}^N} (-1)^{\widetilde{q}\cdot(x\oplus z)}|z\rangle \\
&= \frac{1}{\sqrt{2}}\left(|x\rangle + |\bar{x}\rangle\right) + \frac{1}{2^{N-1}\sqrt{2}} \sum_{\widetilde{q}\in Q_{\mathrm{ew}}} \sum_{z\neq x,\bar{x}} (-1)^{\widetilde{q}\cdot(x\oplus z)}|z\rangle \\
&= \frac{1}{\sqrt{2}}\left(|x\rangle + |\bar{x}\rangle\right).
\end{aligned}$$

Note that the last equality in the above equations holds since the vector $\frac{1}{2^{N-1}\sqrt{2}} \sum_{\widetilde{q}\in Q_{\mathrm{ew}}} \sum_{z\neq x,\bar{x}} (-1)^{\widetilde{q}\cdot(x\oplus z)}|z\rangle$ must vanish by the fact that this vector is orthogonal to the vector $\frac{1}{\sqrt{2}}(|x\rangle + |\bar{x}\rangle)$ that already has a unit length. For any $x \neq y$, the two states

$H^{\otimes N}|\psi_x\rangle = \frac{1}{\sqrt{2}}(|x\rangle + |\overline{x}\rangle)$ and $H^{\otimes N}|\psi_y\rangle = \frac{1}{\sqrt{2}}(|y\rangle + |\overline{y}\rangle)$ are orthogonal since $x \neq \overline{y}$ by the restriction of their Hamming weights. This implies that for any two different x and y, the two states $|\psi_x\rangle$ and $|\psi_y\rangle$ are orthogonal, and hence there is a unitary transformation $W : |x\rangle \mapsto |\psi_x\rangle$. Thus we can design an algorithm similar to Bernstein–Vazirani [5] just replacing the Hadamard transform by W. For a concrete (polynomial-time) construction of W, see Appendix A. □

Now we give the proof of our main result.

Proof of Theorem 1. For exposition, we first give a bounded-error algorithm ($Find^*(k)$) and then make it exact ($Find(k)$). For any query string $\widetilde{q} \in \{0,1\}^N$ with even Hamming weight, the set $P_{\widetilde{q}}$ is defined as

$$P_{\widetilde{q}} := \left\{ q \in Q_{\mathrm{wt}(\widetilde{q})/2} \mid |q_i| = \widetilde{q}_i \text{ for any } i \in \{1,2,\ldots,N\} \right\}.$$

Notice that this corresponds to the set of all "partitions" of the set $\{i \mid \widetilde{q}_i = 1\}$ into two sets of size $\mathrm{wt}(\widetilde{q})/2$, and that each partition in the set, which specifies how to split the $\mathrm{wt}(\widetilde{q})$ coins in half to place them on the left and right pans, can be identified with the corresponding query q in $P_{\widetilde{q}}$. In the rest of this section, for simplicity, we fix the oracle x and denote the answer $\chi_x(q)$ for the query q to x by $\chi(q)$.

Algorithm $Find^*(k)$**.**

1. Prepare N qubits $|0\rangle^{\otimes N}$ in a register R, and apply a unitary transformation W of Lemma 1 to them. Then, we have the state $\frac{1}{\sqrt{2^{N-1}}}\sum_{\widetilde{q}\in Q_{\mathrm{ew}}} |\widetilde{q}\rangle_{\mathsf{R}}$.

2. Conditionally on register R being in state $|\widetilde{q}\rangle$, implement Steps 2.1–2.4 on a register R'.

2.1. Apply a unitary transformation $\mathcal{A}_{\widetilde{q}}$ to the initial state $|0\rangle$ on R' to create a quantum state $\mathcal{A}_{\widetilde{q}}|0\rangle := \sum_{q\in P_{\widetilde{q}}} \alpha|q\rangle_{\mathsf{R}'}$ with $\alpha = \frac{1}{\sqrt{|P_{\widetilde{q}}|}}$, which represents a uniform superposition of all partitions q in $P_{\widetilde{q}}$. Then, the current state is

$$\begin{aligned}|\xi_{2,1}\rangle &= \frac{1}{\sqrt{2^{N-1}}}\sum_{\widetilde{q}\in Q_{\mathrm{ew}}} |\widetilde{q}\rangle_{\mathsf{R}} \sum_{q\in P_{\widetilde{q}}} \alpha|q\rangle_{\mathsf{R}'} \\ &= \frac{1}{\sqrt{2^{N-1}}}\left(\sum_{\widetilde{q}\in Q_{\mathrm{ew}}\cap Q_{\mathrm{efc}}} |\widetilde{q}\rangle_{\mathsf{R}} \sum_{q\in P_{\widetilde{q}}} \alpha|q\rangle_{\mathsf{R}'} + \sum_{\widetilde{q}\in Q_{\mathrm{ew}}\cap Q_{\mathrm{ofc}}} |\widetilde{q}\rangle_{\mathsf{R}} \sum_{q\in P_{\widetilde{q}}} \alpha|q\rangle_{\mathsf{R}'} \right),\end{aligned}$$

where Q_{efc} (resp. Q_{ofc}) denotes the set of all $\widetilde{q}$'s such that the set $\{i \mid \widetilde{q}_i = 1\}$ includes an even (resp. odd) number of false coins.

2.2. Let $\overline{\chi}$ be the Boolean function defined by $\overline{\chi}(q) = 1$ if and only if $\chi(q) = 0$ (that is, the balance scale is balanced). Then, under the above $\mathcal{A}_{\widetilde{q}}$ and $\overline{\chi}$, run the amplitude amplification algorithm **QSearch**$(\mathcal{A}_{\widetilde{q}}, \overline{\chi})$ when the initial success probability of $\mathcal{A}_{\widetilde{q}}$ is unknown (Theorem 3 in [7]). Here "success" means that the balance scale is balanced, and hence we use $\overline{\chi}$, not χ, in **QSearch**. Then we obtain a state in the form of

$$|\xi_{2,2}\rangle = \frac{1}{\sqrt{2^{N-1}}}\left(\sum_{\widetilde{q}\in Q_{\mathrm{ew}}\cap Q_{\mathrm{efc}}} |\widetilde{q}\rangle_{\mathsf{R}} \sum_{q\in P_{\widetilde{q}}} \beta_q|q,g_q\rangle_{\mathsf{R}'} + \sum_{\widetilde{q}\in Q_{\mathrm{ew}}\cap Q_{\mathrm{ofc}}} |\widetilde{q}\rangle_{\mathsf{R}} \sum_{q\in P_{\widetilde{q}}} \alpha|q,g_q\rangle_{\mathsf{R}'} \right)$$

where $|g_q\rangle$ is a garbage state. Note that, in the first term, the amplitudes β_q such that $\overline{\chi}(q) = 1$ are now large by amplitude amplification while the second term does not change since the balance scale is always tilted.

2.3. If Step 2.2 finds a "solution", i.e., a partition q such that $\overline{\chi}(q) = 1$, then do nothing. Otherwise, flip the phase (and then the phase is kick-backed into R). Notice that when $\{i \mid \widetilde{q}_i = 1\}$ includes an odd number of false coins, the phase is always flipped, while when it includes an even number of false coins, the phase is not flipped with high amplitude. Now the current state is

$$\begin{aligned}|\xi_{2,3}\rangle &= \frac{1}{\sqrt{2^{N-1}}}\left(\sum_{\widetilde{q}\in Q_{\mathrm{ew}}\cap Q_{\mathrm{efc}}} |\widetilde{q}\rangle_{\mathsf{R}} \sum_{q\in P_{\widetilde{q}}} \beta_q(-1)^{\chi(q)}|q,g_q\rangle_{\mathsf{R}'} - \sum_{\widetilde{q}\in Q_{\mathrm{ew}}\cap Q_{\mathrm{ofc}}} |\widetilde{q}\rangle_{\mathsf{R}} \sum_{q\in P_{\widetilde{q}}} \alpha|q,g_q\rangle_{\mathsf{R}'} \right) \\ &= \frac{1}{\sqrt{2^{N-1}}}\left(\sum_{\widetilde{q}\in Q_{\mathrm{ew}}\cap Q_{\mathrm{efc}}} |\widetilde{q}\rangle_{\mathsf{R}} \sum_{q\in P_{\widetilde{q}}} \beta_q|q,g_q\rangle_{\mathsf{R}'} - \sum_{\widetilde{q}\in Q_{\mathrm{ew}}\cap Q_{\mathrm{ofc}}} |\widetilde{q}\rangle_{\mathsf{R}} \sum_{q\in P_{\widetilde{q}}} \alpha|q,g_q\rangle_{\mathsf{R}'} \right) \\ &\quad - 2\sum_{\widetilde{q}\in Q_{\mathrm{ew}}\cap Q_{\mathrm{efc}}} |\widetilde{q}\rangle_{\mathsf{R}}|err_{\widetilde{q}}\rangle_{\mathsf{R}'}\end{aligned}$$

where $|err_{\widetilde{q}}\rangle_{\mathsf{R}'} = \frac{1}{\sqrt{2^{N-1}}}\sum_{q\in P_{\widetilde{q}}:\chi(q)=1} \beta_q|q,g_q\rangle_{\mathsf{R}'}$.

2.4. Reverse the quantum transformation done in Steps 2.1 and 2.2. Notice that the reversible transformation is done on R′ in parallel for each $\widetilde{q}$ while the contents of R does not change since it is the control part. Therefore, the state becomes

$$|\xi_{2,4}\rangle = \frac{1}{\sqrt{2^{N-1}}} \sum_{\widetilde{q}\in Q_{ew}\cap Q_{efc}} |\widetilde{q}\rangle_{R} |0\rangle_{R'} - \frac{1}{\sqrt{2^{N-1}}} \sum_{\widetilde{q}\in Q_{ew}\cap Q_{ofc}} |\widetilde{q}\rangle_{R} |0\rangle_{R'} - 2 \sum_{\widetilde{q}\in Q_{ew}\cap Q_{efc}} |\widetilde{q}\rangle_{R} |err'_{\widetilde{q}}\rangle_{R'}$$
$$= \frac{1}{\sqrt{2^{N-1}}} \sum_{\widetilde{q}\in Q_{ew}} (-1)^{\widetilde{q}\cdot x} |\widetilde{q}\rangle_{R} |0\rangle_{R'} - 2 \sum_{\widetilde{q}\in Q_{ew}\cap Q_{efc}} |\widetilde{q}\rangle_{R} |err'_{\widetilde{q}}\rangle_{R'}$$

where $|err'_{\widetilde{q}}\rangle_{R'}$ is the transformed state of $|err_{\widetilde{q}}\rangle_{R'}$.

3. Apply W^{-1} to the state in R. By Lemma 1, we obtain the final state

$$|\xi_3\rangle = |x\rangle_{R} |0\rangle_{R'} - 2W^{-1} \left(\sum_{\widetilde{q}\in Q_{ew}\cap Q_{efc}} |\widetilde{q}\rangle_{R} |err'_{\widetilde{q}}\rangle_{R'} \right).$$

Then measure R in the computational basis. (End of Algorithm)

For justifying the correctness of *Find**(*k*), it suffices to show that the squared magnitude of the second term of $|\xi_3\rangle$ is a small constant, say, 1/400, since we then measure the desired value *x* with probability at least 9/10 (in fact, at least $(1-\sqrt{1/400})^2 > 9/10$). By the unitarity, its squared magnitude is equal to that of the last term of $|\xi_{2,3}\rangle$. Therefore, we evaluate the following value ϵ.

$$\epsilon := 4 \left\| \sum_{\widetilde{q}\in Q_{ew}\cap Q_{efc}} |\widetilde{q}\rangle_{R} |err_{\widetilde{q}}\rangle_{R'} \right\|^2 = \frac{4}{2^{N-1}} \sum_{\widetilde{q}\in Q_{ew}\cap Q_{efc}} \left\| \sum_{q\in P_{\widetilde{q}}:\chi(q)=1} \beta_q |q, g_q\rangle_{R'} \right\|^2.$$

Lemma 2. *ϵ is at most* 1/400.

Proof. Consider an arbitrary $\widetilde{q}$ in $Q_{ew}\cap Q_{efc}$ (i.e., it involves an even number of false coins). When $\{i \mid \widetilde{q}_i = 1\}$ includes *m* ($\le k$) false coins (where *m* is even), the probability that a partition *q* is such that $\overline{\chi}(q) = 1$ is at least

$$p = \frac{\binom{m}{m/2}\binom{\mathrm{wt}(\widetilde{q})-m}{(\mathrm{wt}(\widetilde{q})-m)/2}}{\binom{\mathrm{wt}(\widetilde{q})}{\mathrm{wt}(\widetilde{q})/2}} = \Omega\left(\frac{1}{\sqrt{m}}\right) = \Omega\left(\frac{1}{\sqrt{k}}\right).$$

By Theorem 3 in [7], it is guaranteed that, in the algorithm **QSearch**($\mathcal{A}_{\widetilde{q}}$, $\overline{\chi}$), an expected number of applications of the Grover-like subroutine to find a "solution", i.e., a partition *q* such that $\overline{\chi}(q) = 1$, is bounded by $O(1/\sqrt{p}) = O(k^{1/4})$. The subroutine consists of (i) $\mathcal{A}_{\widetilde{q}}$, (ii) its inverse, (iii) the transformation $O_{\overline{\chi}}$ defined by $O_{\overline{\chi}}|q\rangle = (-1)^{\overline{\chi}(q)}|q\rangle$, and (iv) the transformation U_0 defined by $U_0|z\rangle = |z\rangle$ if $z \neq 0$ and $-|z\rangle$ if $z = 0$. Note that $\mathcal{A}_{\widetilde{q}}$ (and hence its inverse) and U_0 can be implemented without any query to the *B*-oracle, and $O_{\overline{\chi}}$ can be implemented with one query to the *B*-oracle. Thus the expected number of queries to find a "solution" is $O(k^{1/4})$. By setting the number of applications of the subroutine to $c_0 k^{1/4}$ where c_0 is a large constant, Step 2.2 finds a "solution" with probability at least 1599/1600. This means that for any $\widetilde{q}$ in $Q_{ew}\cap Q_{efc}$, $\sum_{q\in P_{\widetilde{q}}:\overline{\chi}(q)=0} \beta_q |q, g_q\rangle_{R'}$ has squared magnitude at most 1/1600. Thus we have

$$\epsilon = \frac{4}{2^{N-1}} \sum_{\widetilde{q}\in Q_{ew}\cap Q_{efc}} \left\| \sum_{q\in P_{\widetilde{q}}:\overline{\chi}(q)=0} \beta_q |q, g_q\rangle_{R'} \right\|^2 \le \frac{1}{400}.$$

This completes the proof of Lemma 2. □

Finally, it is easy to see from the above proof that the query complexity of *Find**(*k*) is $O(k^{1/4})$ since it makes $O(k^{1/4})$ queries in Step 2 and no queries in Steps 1 and 3.

Now we consider the exact algorithm *Find*(*k*). By the symmetric structure of algorithm *Find**(*k*), the success probability of identifying *x* correctly is independent of *x* (recall that the oracle candidates are $\binom{N}{k}$ *N*-bit strings *x* with Hamming weight *k*). Thus we can use the so-called exact amplitude amplification algorithm (Theorem 4 in [7]) to convert it into the exact algorithm.

Here is the brief description of *Find*(*k*) (see Appendix B for the details). First, we implement *Find**(*k*). As shown above, *Find**(*k*) produces the correct output (i.e., the *k* false coins) with a constant probability ($\ge 9/10$) larger than 1/4. Notice that we can make the success probability exactly 1/4 by an appropriate adjustment. Second, we need an algorithm for checking if the output is correct to amplify the success probability to 1. Namely, an algorithm *Check* needs to judge whether *k* coins are

indeed all false, which can be implemented classically in $O(\log k)$ queries (as seen in Appendix B). Then we can implement the exact amplitude amplification: like the 1/4-Grover's algorithm [6], flip the phase if *Check* judges that the output is correct, and apply the reflection about the state obtained after $Find^*(k)$. It is not difficult to see that $Find(k)$ always finds the k false coins and the total complexity is $O(k^{1/4})$. Therefore, the proof of Theorem 1 is completed. □

3. Lower bounds

This section discusses the lower bounds for the query complexity of finding the k false coins from N coins (i.e., the oracle $x \in \{0,1\}^N$ with $\mathrm{wt}(x) = k$). We conjecture that the upper bound $O(k^{1/4})$ is tight but, unfortunately, we have not been able to show whether it is true or not. Instead, we show that if there would be an algorithm that improves the upper bound essentially, then it would have a completely different structure from our algorithm.

3.1. Basic ideas

Before describing our results, we observe two properties of our algorithm $Find(k)$. First, $Find(k)$ uses only "big pans", i.e., it always places at least $\Omega(N)$ coins on the pans, which is called the *big-pan property*. (The algorithm $Find^*(k)$ in Section 2 uses "small pans" but it can be adapted with no essential change so that it works even if the size of pans must be big, as easily shown in Appendix B.) Second, the B-oracle is always used in such a way that once the coins placed on the two pans are determined, the partition of them into the two pans is done uniformly at random, which is called the *random-partition property*. In this section, we show two $\Omega(k^{1/4})$ lower bounds: the first one holds for any algorithm that satisfies the big-pan property. The second one also indicates that if our algorithm has the random-partition property, then it is unlikely to beat the $O(k^{1/4})$ upper bound.

For this purpose, we revisit one version of the (nonnegative) quantum adversary method, called *the strong weighted adversary method* in [21], due to Zhang [23]. Let f be a function from a finite set S to another finite set S'. Recall that in a query complexity model, an input x in S is given as an oracle. An algorithm $\mathcal{A}$ would like to compute $f(x)$ while it can obtain the information about x by a unitary transformation

$$O_x|q, a, z\rangle = |q, a \oplus \zeta_x(q), z\rangle, \tag{1}$$

where $|q\rangle$ is the register for a query string q from a finite set Q, $|a\rangle$ is the register for the binary answer $\zeta_x(q)$, and $|z\rangle$ is the work register. Note that the adversary method usually assumes the so-called index oracle, namely x is a string in $\{0,1\}^N$, q is an integer $1 \le i \le N$, and $\zeta_x(q)$ is the ith bit (0 or 1) of x. However, one can easily see that the above generalization to $\zeta_x(q)$ requires no essential changes for its proof. Thus Theorem 14 of [23] can be restated as follows:

Lemma 3. *Let w, w' denote a weight scheme as follows:*

1. *Every pair $(x, y) \in S \times S$ is assigned a nonnegative weight $w(x, y) = w(y, x)$ that satisfies $w(x, y) = 0$ whenever $f(x) = f(y)$.*
2. *Every triple $(x, y, q) \in S \times S \times Q$ is assigned a nonnegative weight $w'(x, y, q)$ that satisfies $w'(x, y, q) = 0$ whenever $\zeta_x(q) = \zeta_y(q)$ or $f(x) = f(y)$, and $w'(x, y, q)w'(y, x, q) \ge w^2(x, y)$ for all x, y, q such that $\zeta_x(q) \ne \zeta_y(q)$ and $f(x) \ne f(y)$.*

For all x, q, let $\mu(x) = \sum_y w(x, y)$ and $\nu(x, q) = \sum_y w'(x, y, q)$. Then, the quantum query complexity of f is at least

$$\Omega\left(\max_{w,w'} \min_{\substack{x,y,q:\ w(x,y)>0,\\ \zeta_x(q)\ne\zeta_y(q)}} \sqrt{\frac{\mu(x)\mu(y)}{\nu(x,q)\nu(y,q)}}\right).$$

3.2. Big pan lower bounds

First, we show that our upper bound is tight under the big-pan property. In what follows, $L \ge l$ denotes the restriction that at least l coins must be placed on each pan whenever the balance scale is used.

Theorem 2. *If $L \ge l$, any quantum algorithm needs $\Omega((lk/N)^{1/4})$ queries to find the k false coins among N coins. In particular, $\Omega(k^{1/4})$ queries are necessary if there is some constant c such that $L \ge N/c$.*

Proof. Let $l = N/d$. The lower bound we should show is $\Omega((k/d)^{1/4})$. We can assume that $d \le k/3$ (otherwise, the lower bound becomes trivial). To use Lemma 3, let $S = \{x \in \{0,1\}^N \mid \mathrm{wt}(x) = k\}$, $Q = Q_{\ge N/d}$, $\zeta_x(q) = \chi_x(q)$, and $f(x) = x$, where the set $Q_{\ge l'}$ is defined as $Q_{\ge l'} = \bigcup_{l \ge l'} Q_l$. Then, our weight scheme is as follows: let $w(x, y) = 1$ for any pair $(x, y) \in S \times S$ such that $x \ne y$, and let $w'(x, y, q) = 1$ for all $(x, y, q) \in S \times S \times Q_{\ge N/d}$ such that $\chi_x(q) \ne \chi_y(q)$ and $x \ne y$. It is easy to check that this satisfies the condition of a weight scheme. Then, for any x, we have $\mu(x) = \sum_y w(x, y) = \binom{N}{k} - 1$. We need to evaluate $\nu(x, q)\nu(y, q)$ for pairs (x, y) such that $\chi_x(q) = 1$ and $\chi_y(q) = 0$ or $\chi_x(q) = 0$ and $\chi_y(q) = 1$. By symmetry, it suffices to consider the case where $\chi_x(q) = 1$ and $\chi_y(q) = 0$. Fix $q \in Q_{\ge N/d}$ arbitrarily and assume that $q \in Q_{N/c}$ where $c \le d$. Since $\chi_x(q) = 1$ means that for input x the balance scale is tilted when N/c coins are placed on each pan according

to a query q, we can see that $\nu(x,q) = \sum_y w'(x,y,q)$ is the number of all inputs y's such that the balance scale is balanced for the same query q. Therefore, by summing up all the cases such that those N/c coins include m false ones, it holds that

$$\nu(x,q) = \gamma(N,k,c),$$

where $\gamma(N,k,c)$ is defined as

$$\gamma(N,k,c) = \sum_{m=0}^{k/2} \binom{N/c}{m}^2 \binom{(1-2/c)N}{k-2m}.$$

Since $\chi_y(q) = 0$, we have $\nu(y,q) = \sum_x w'(x,y,q) = \binom{N}{k} - \gamma(N,k,c)$ by counting all x's such that the balance scale is titled. Then the product $\nu(x,q)\nu(y,q)$ is $\gamma(N,k,c)\left(\binom{N}{k} - \gamma(N,k,c)\right)$. By Lemma 3 the quantum query complexity of our problem is at least

$$\Omega\left(\min_{c:\, c\le d} \sqrt{\frac{\left(\binom{N}{k}-1\right)^2}{\gamma(N,k,c)\left(\binom{N}{k}-\gamma(N,k,c)\right)}}\right) = \Omega\left(\min_{c:\, c\le d} \sqrt{\frac{\binom{N}{k}}{\gamma(N,k,c)}}\right). \tag{2}$$

Then, we can show the following lemma.

Lemma 4. *$\gamma(N,k,c)/\binom{N}{k} = O(\sqrt{c/k})$ for any $2 \le c \le d\ (\le k/3)$.*

Proof. Note that $\gamma(N,k,c)/\binom{N}{k}$ means the probability that the balance scale is balanced when N/c coins (N coins include k false ones) are randomly placed on each pan, and hence its value decreases as c approaches to 2. So, it suffices to prove the lemma for $c \ge 4$.

Let us denote each term in the sum $\gamma(N,k,c)$ by $t(m) = \binom{N/c}{m}^2\binom{(1-2/c)N}{k-2m}$ for $m = 0, 1, \ldots, k/2$. We divide $\gamma(N,k,c)$ into the two parts, that is, we write $\gamma(N,k,c) = T_{>k/2c} + T_{\le k/2c}$ where $T_{>k/2c} = \sum_{m:m>k/2c} t(m)$ and $T_{\le k/2c} = \sum_{m:m\le k/2c} t(m)$. For the proof, it suffices to show that both $T_{>k/2c}/\binom{N}{k}$ and $T_{\le k/2c}/\binom{N}{k}$ are bounded by $O(\sqrt{c/k})$. First we consider $T_{>k/2c}/\binom{N}{k}$. When N/c coins are randomly placed on each pan, let E_1 be the event that at least k/c false coins are placed on the pans, and E_2 be the event that the balance scale is balanced. Then, we can see that

$$\begin{aligned}\frac{T_{>k/2c}}{\binom{N}{k}} &= \Pr[E_1 \wedge E_2] \\ &\le \Pr[E_2|E_1] = O(1/\sqrt{k/c}) \\ &= O(\sqrt{c/k}).\end{aligned}$$

Second we consider $T_{\le k/2c}/\binom{N}{k}$. Let $r(m) = t(m+1)/t(m)$. Note that $r(m)$ is monotone decreasing on m since

$$\begin{aligned} r(m) &= \frac{\binom{N/c}{m+1}^2\binom{(1-2/c)N}{k-2(m+1)}}{\binom{N/c}{m}^2\binom{(1-2/c)N}{k-2m}} \\ &= \frac{\left(\frac{N}{c}-m\right)^2(k-2m)(k-2m-1)}{(m+1)^2((1-2/c)N-k+2m+1)((1-2/c)N-k+2m+2)}. \end{aligned}$$

Now we verify that $r(k/2c-1) > 4$. In fact, since $c \le k/3 < 2 + k/2$, we have

$$(1-2/c)N - k + k/c < (1-2/c)(N-k/2-c) \tag{3}$$

and

$$k - k/c - 3 \ge k(1-2/c). \tag{4}$$

Thus we obtain

$$\begin{aligned} r(k/2c-1) &= \frac{(1/c)^2(N-k/2-c)^2(k-k/c-2)(k-k/c-3)}{(k/2c)^2((1-2/c)N-k+k/c)((1-2/c)N-k+k/c-1)} \\ &> \frac{4(k-k/c-2)(k-k/c-3)}{k^2(1-2/c)^2} \quad \text{(by Eq. (3))} \\ &\ge 4 \quad \text{(by Eq. (4))}. \end{aligned}$$

These facts imply that

$$T_{\le k/2c} = \sum_{m:m\le k/2c} t(m) < \left(1 + 1/4 + (1/4)^2 + \cdots\right) t(k/2c) = (4/3)t(k/2c),$$

which is bounded by $(4/3)t(k/c)$ since $t(m)$ takes the maximum value at $m = k/c$. Calculating $t(k/c)/\binom{N}{k}$ using the Stirling formula $n! \sim \sqrt{2\pi n}(N/e)^N$, we obtain

$$\frac{t(k/c)}{\binom{N}{k}} = \frac{\binom{N/c}{k/c}^2\binom{(1-2/c)N}{(1-2/c)k}}{\binom{N}{k}} = \frac{\frac{k!}{((\frac{k}{c})!)^2((1-\frac{2}{c})k)!}\cdot\frac{(N-k)!}{((\frac{N-k}{c})!)^2((1-\frac{2}{c})(N-k))!}}{\frac{N!}{((\frac{N}{c})!)^2((1-\frac{2}{c})N)!}}$$
$$\sim \frac{cN}{2\pi k(N-k)\sqrt{1-2/c}},$$

which is bounded by $O(c/k)$ since $k \le N/2$ and $c \ge 4$. Thus, the sum $T_{\le k/2c}/\binom{N}{k}$ is bounded by $O(c/k) = O(\sqrt{c/k})$. From the above, we obtain $\gamma(N,k,c)/\binom{N}{k} = O(\sqrt{c/k})$. □

Now Lemma 4 implies the desired bound $\Omega((k/d)^{1/4})$ by Eq. (2), and hence the proof of Theorem 2 is completed. □

On the contrary, we can show that any algorithm that uses only "small pans" also needs $\Omega(k^{1/4})$ queries (Theorem 5). For instance, we cannot break the current bound $k^{1/4}$ by any algorithm that places $O(N/k)$ coins on the pans. (Notice that the pans include only a constant number of false coins with high probability in this case and therefore the balance scale can be balanced with a better probability for the case where the pans include an even number of false coins, but at the same time, we cannot use a wide range of superpositions.) Moreover, we can obtain another lower bound for the case where "big pans" and "small pans" are both available but "medium pans" are not (Theorem 6). Unfortunately, one can see that there is still a gap between the sizes of the big pans and small pans even for a weakest nontrivial ($\omega(1)$) lower bound. See Appendix C for the details of these results.

3.3. Lower bounds for the quasi B-oracle

Recall that we wish to prove that our upper bound is tight under the random partition property. Unfortunately, there is no obvious way of doing that since the property is on the "structure of the algorithm", not a restriction on the oracle itself. Instead, our approach is to introduce a new oracle (what we call the quasi B-oracle) that seems both to inherit the nature of the B-oracle and to extract an essence of the random partition property.

We generalize the "deterministic" oracles O_x given by Eq. (1) in the following "stochastic" form: For any $x \in S$, a stochastic oracle $\widetilde{O}_x$ is defined as

$$\widetilde{O}_x|q,a,z\rangle = \sqrt{\Pr[\zeta_x(q)=0]}|q,a,z\rangle + (-1)^a\sqrt{\Pr[\zeta_x(q)=1]}|q,a\oplus 1,z\rangle, \tag{5}$$

where the probabilities $\Pr[\zeta_x(q)=0]$ and $\Pr[\zeta_x(q)=1]$ are taken over the "coin toss" of the stochastic oracle whose bias is determined as a function of x and q (we should be careful not to lose its unitarity). Then, we can define the *quasi B-oracle* by setting

$$\Pr[\zeta_x(q)=0] = \begin{cases} 0 & (\text{if } \mathrm{wt}(x\wedge q) \text{ is odd}) \\ \sqrt{1/\mathrm{wt}(x\wedge q)} & (\text{if } \mathrm{wt}(x\wedge q) \text{ is positive and even}) \\ 1 & (\text{if } \mathrm{wt}(x\wedge q)=0), \end{cases} \tag{6}$$

where x and q are N-bit strings, and $x\wedge q$ is the N-bit string obtained by the bitwise AND of x and q. The quasi B-oracle reflects the following fact under the random-partition property: if the coins on the pans include an odd number of false ones, then the balance scale is always tilted, and if they include an even number ($=m$) of false ones, the balance scale will be balanced with probability $1/\sqrt{m}$. (Note that the bit strings q in Eq. (6) corresponds to the bit strings $\widetilde{q}$ in *Find*$^*(k)$.)

Here we generalize Lemma 3 to the case of stochastic oracles.

Lemma 5. *Let w, w' denote a weight scheme as Lemma 3 except replacing Condition 2 to*

2′ *Every triple $(x,y,q) \in S\times S\times Q$ is assigned a nonnegative weight $w'(x,y,q)$ that satisfies $w'(x,y,q) = 0$ whenever $\Pr[\zeta_x(q)=\zeta_y(q)]=1$ or $f(x)=f(y)$, and $w'(x,y,q)w'(y,x,q) \ge w^2(x,y)$ for all x,y,q such that $\Pr[\zeta_x(q)\ne\zeta_y(q)]>0$ and $f(x)\ne f(y)$.*

Then, the quantum query complexity of f for the stochastic oracle given in Eq. (5) is at least

$$\Omega\left(\max_{w,w'}\min_{\substack{x,y,q:\ w(x,y)>0,\\ \Pr[\zeta_x(q)\ne\zeta_y(q)]>0}}\sqrt{\frac{\mu(x)\mu(y)}{\nu(x,q)\nu(y,q)}}\frac{1}{\sqrt{P_{01,q}}+\sqrt{P_{10,q}}}\right),$$

where $P_{ab,q} = \Pr[\zeta_x(q)=a]\Pr[\zeta_y(q)=b]$.

Proof. The proof follows that of [23, Theorem 14] essentially; in the following we mainly describe the difference. Assume that there is a T-query quantum algorithm $\mathcal{A}$ computing f with high probability. Note that the initial state of $\mathcal{A}$ is $|\psi_x^0\rangle = |0\rangle$

for any input x. The final state for input x can be written as $|\psi_x^T\rangle = U_{T-1}\widetilde{O}_x \cdots U_1\widetilde{O}_x U_0|0\rangle$ for some unitary transformations $U_0, \ldots, U_{T-1}$. Since $\mathcal{A}$ computes f with high probability, there is some constant $\epsilon < 1$ such that $|\langle\psi_x^T|\psi_y^T\rangle| \le \epsilon$ for any x and y with $f(x) \neq f(y)$. Let $|\psi_x^k\rangle = U_{k-1}\widetilde{O}_x \cdots U_1\widetilde{O}_x U_0|0\rangle$. For any x and y with $f(x) \neq f(y)$, we can represent

$$|\psi_x^{k-1}\rangle = \sum_{q,a,z} \alpha_{q,a,z}|q, a, z\rangle, \qquad |\psi_y^{k-1}\rangle = \sum_{q,a,z} \beta_{q,a,z}|q, a, z\rangle.$$

After querying to the oracle, we have

$$\begin{aligned}
\widetilde{O}_x|\psi_x^{k-1}\rangle &= \sum_{q,a,z} \alpha_{q,a,z}\left(\sqrt{\Pr[\zeta_x(q)=0]}|q, a, z\rangle + (-1)^a\sqrt{\Pr[\zeta_x(q)=1]}|q, a\oplus 1, z\rangle\right)\\
&= \sum_{q,a,z}\left(\sqrt{\Pr[\zeta_x(q)=0]}\alpha_{q,a,z} + (-1)^{a\oplus 1}\sqrt{\Pr[\zeta_x(q)=1]}\alpha_{q,a\oplus 1,z}\right)|q, a, z\rangle,\\
\widetilde{O}_y|\psi_y^{k-1}\rangle &= \sum_{q,a,z}\left(\sqrt{\Pr[\zeta_y(q)=0]}\beta_{q,a,z} + (-1)^{a\oplus 1}\sqrt{\Pr[\zeta_y(q)=1]}\beta_{q,a\oplus 1,z}\right)|q, a, z\rangle.
\end{aligned}$$

Hence we have (recall that $P_{ab,q} := \Pr[\zeta_x(q)=a]\Pr[\zeta_y(q)=b]$):

$$\begin{aligned}
\langle\psi_x^k|\psi_y^k\rangle &= \sum_{q,a,z}\sqrt{P_{00,q}}\alpha^*_{q,a,z}\beta_{q,a,z} + \sum_{q,a,z}\sqrt{P_{11,q}}\alpha^*_{q,a\oplus 1,z}\beta_{q,a\oplus 1,z}\\
&\quad + \sum_{q,a,z}(-1)^{a\oplus 1}\sqrt{P_{01,q}}\alpha^*_{q,a,z}\beta_{q,a\oplus 1,z} + \sum_{q,a,z}(-1)^{a\oplus 1}\sqrt{P_{10,q}}\alpha^*_{q,a\oplus 1,z}\beta_{q,a,z}\\
&= \sum_{q,a,z}\sqrt{P_{00,q}}\alpha^*_{q,a,z}\beta_{q,a,z} + \sum_{q,a,z}\sqrt{P_{11,q}}\alpha^*_{q,a,z}\beta_{q,a,z}\\
&\quad + \sum_{q,a,z}(-1)^{a\oplus 1}\sqrt{P_{01,q}}\alpha^*_{q,a,z}\beta_{q,a\oplus 1,z} + \sum_{q,a,z}(-1)^{a}\sqrt{P_{10,q}}\alpha^*_{q,a,z}\beta_{q,a\oplus 1,z}.
\end{aligned}$$

On the contrary,

$$\langle\psi_x^{k-1}|\psi_y^{k-1}\rangle = \sum_{q,a,z}\alpha^*_{q,a,z}\beta_{q,a,z}.$$

Thus the difference between $\langle\psi_x^{k-1}|\psi_y^{k-1}\rangle$ and $\langle\psi_x^k|\psi_y^k\rangle$ is

$$\begin{aligned}
\langle\psi_x^{k-1}|\psi_y^{k-1}\rangle - \langle\psi_x^k|\psi_y^k\rangle = \sum_{q,a,z:\Pr[\zeta_x(q)\neq\zeta_y(q)]>0}\Big[&\left(1-\sqrt{P_{00,q}}-\sqrt{P_{11,q}}\right)\alpha^*_{q,a,z}\beta_{q,a,z}\\
&+(-1)^a\left(\sqrt{P_{01,q}}\alpha^*_{q,a,z}\beta_{q,a\oplus 1,z} - \sqrt{P_{10,q}}\alpha^*_{q,a,z}\beta_{q,a\oplus 1,z}\right)\Big]
\end{aligned}$$

since $\Pr[\zeta_x(q) = \zeta_y(q)] = 1$, that is, $P_{00,q} + P_{11,q} = 1$ implies that $P_{00,q} = 1$ or $P_{11,q} = 1$. By the triangle inequality,

$$\begin{aligned}
1-\epsilon \le 1 - |\langle\psi_x^T|\psi_y^T\rangle| &\le \sum_{k=1}^{T}|\langle\psi_x^{k-1}|\psi_y^{k-1}\rangle - \langle\psi_x^k|\psi_y^k\rangle|\\
&\le \sum_{k=1}^{T}\sum_{\substack{q,a,z\\ \Pr[\zeta_x(q)\neq\zeta_y(q)]>0}}\left[\left(1-\sqrt{P_{00,q}}-\sqrt{P_{11,q}}\right)|\alpha_{q,a,z}||\beta_{q,a,z}| + \left(\sqrt{P_{01,q}}+\sqrt{P_{10,q}}\right)|\alpha_{q,a,z}||\beta_{q,a\oplus 1,z}|\right]\\
&\le \sum_{k=1}^{T}\sum_{\substack{q,a,z\\ \Pr[\zeta_x(q)\neq\zeta_y(q)]>0}}\left[\left(\sqrt{P_{01,q}}+\sqrt{P_{10,q}}\right)\left(|\alpha_{q,a,z}||\beta_{q,a,z}| + |\alpha_{q,a,z}||\beta_{q,a\oplus 1,z}|\right)\right].
\end{aligned}$$

The remaining part is completely similar to the proof of [23, Theorem 14]. Summing up the inequalities for all $(x, y) \in S\times S$ with weight $w(x, y)$, we have $(1-\epsilon)\sum_{x,y} w(x, y) \le 2T\frac{1}{\sqrt{A}}\sum_{x,y} w(x, y)$ where

$$A = \min_{\substack{x,y,q:\ w(x,y)>0\\ \Pr[\zeta_x(q)\neq\zeta_y(q)]>0}} \frac{\mu(x)\mu(y)}{\nu(x, q)\nu(y, q)}\frac{1}{\left(\sqrt{P_{01,q}}+\sqrt{P_{10,q}}\right)^2}.$$

Therefore, we obtain $T = \Omega(\sqrt{A})$ and hence the proof is completed. □

Now we are ready to give the upper and lower bounds for the query complexity of finding the oracle in case of the quasi B-oracle. Assume that $\mathrm{wt}(x) = k$. The upper bound is easy by modifying Theorem 1 so that Step 2 in *Find*$^*(k)$ can be replaced with $O(k^{1/4})$ repetitions of the quasi B-oracle.

Theorem 3. *There is an $O(k^{1/4})$-query quantum algorithm to find x using the quasi B-oracle.*

On the contrary, we can obtain the tight lower bound by using Lemma 5. The weight scheme contrasts with that of Theorem 2; $w(x, y)$ is nonzero only if the Hamming distance between x and y, denoted as $d(x, y)$, is 2.

Theorem 4. *Any quantum algorithm with the quasi B-oracle needs $\Omega(k^{1/4})$ queries to find x.*

Proof. First we define a weight scheme. Let $S = \{x \in \{0,1\}^N \mid \mathrm{wt}(x) = k\}$ and $f(x) = x$. In what follows, we assume that $\mathrm{wt}(q) = l$ for a query string q that provides the minimum value of the formula of Lemma 5, and show that the theorem holds for an arbitrary $l \le N$. For any $(x, y) \in S \times S$, let $w(x, y) = 1$ if $d(x, y) = 2$ and 0 otherwise. We must satisfy $w'(x, y, q) = 0$ for any different x, y such that $d(x, y) \neq 2$ or $\Pr[\zeta_x(q) = \zeta_y(q)] = 1$, which implies $\mathrm{wt}(x \wedge q) = \mathrm{wt}(y \wedge q)$. Thus we let $w'(x, y, q) \neq 0$ only if $d(x, y) = 2$ and $\mathrm{wt}(x \wedge q) = \mathrm{wt}(y \wedge q) \pm 1$. Define $w'(x, y, q)$ as a function of $\mathrm{wt}(x \wedge q) = m_1$ and $\mathrm{wt}(y \wedge q) = m_2$, and thus denote it by $w'(x, y, q) = w'(m_1, m_2)$. Then $w'(m_1, m_2)$ is taken as

$$w'(m_1, m_2) = \begin{cases} \dfrac{2m(N-k-l+2m)}{(l-2m+1)(k-2m+1)} & \text{if } (m_1, m_2) = (2m-1, 2m) \\ \dfrac{(l-2m+1)(k-2m+1)}{2m(N-k-l+2m)} & \text{if } (m_1, m_2) = (2m, 2m-1) \\ 1 & \text{if } (m_1, m_2) = (2m, 2m+1), (2m+1, 2m) \\ 0 & \text{otherwise.} \end{cases}$$

It can be easily seen that w, w' is a weight scheme. Now we evaluate the lower bound under this weight scheme. Clearly, $\mu(x) = \mu(y) = k(N-k)$. For evaluating $\nu(x, q)\nu(y, q)$, we consider only the case where $m_1 = \mathrm{wt}(x \wedge q) = 2m$ and $m_2 = \mathrm{wt}(y \wedge q) = 2m-1$ (the other cases such as $m_1 = 2m$ and $m_2 = 2m+1$ can be similarly analyzed). In this case, we have

$$\begin{aligned} \nu(x, q) &= 2m(N-l-k+2m)w'(2m, 2m-1) + (k-2m)(l-2m)w'(2m, 2m+1) \\ &\le (l-2m+1)(k-2m+1) + (k-2m)(l-2m) \\ &\le 2(l-2m+1)(k-2m+1), \\ \nu(y, q) &= (2m-1)(N-k-l+2m-1)w'(2m-1, 2m-2) + (k-2m+1)(l-2m+1)w'(2m-1, 2m) \\ &\le (2m-1)(N-k-l+2m-1) + 2m(N-k-l+2m) \\ &\le 4m(N-k-l+2m). \end{aligned}$$

Note that $P_{01,q} = 1/\sqrt{2m}$ and $P_{10,q} = 0$ since $\Pr[\zeta_x(q) = 0] = \sqrt{1/2m}$ and $\Pr[\zeta_y(q) = 1] = 1$. Thus we have

$$\frac{\mu(x)\mu(y)}{\nu(x, q)\nu(y, q)} \frac{1}{(\sqrt{P_{01,q}} + \sqrt{P_{10,q}})^2} = \frac{k^2(N-k)^2\sqrt{2m}}{8m(l-2m+1)(k-2m+1)(N-k-l+2m)}.$$

This value is bounded below by $\Omega(k^{1/2})$ since $m \le k/2$ and $l \le N$. Now Lemma 5 completes the proof. □

Acknowledgments

We are grateful to Mario Szegedy for directing our interest to the topic of this paper, and an anonymous referee for a helpful idea to improve the earlier upper bounds for general k significantly. We are also grateful to Seiichiro Tani and Shigeru Yamashita for helpful discussions.

Appendix A. Efficient construction of transformation W

It can be easily seen that our algorithm $Find^*(k)$ can be implemented in time polynomial in the length of the input except for a bit nontrivial task, constructing the transformation W in the proof of Lemma 1. Recall that W is a unitary transformation that satisfies $W|x\rangle = |\psi_x\rangle$ for any $x \in S_{<N/2} = \{x \in \{0,1\}^N \mid \mathrm{wt}(x) < N/2\}$, where $|\psi_x\rangle = \frac{1}{\sqrt{2^{N-1}}} \sum_{\widetilde{q} \in Q_{\mathrm{ew}}} (-1)^{\widetilde{q} \cdot x} |\widetilde{q}\rangle$. We define a subset $S^+_{<N/2}$ of size $2^N/2$ as follows: $S^+_{<N/2} = S_{<N/2}$ if N is odd, or $S^+_{<N/2} = S_{<N/2} \cup \{x \in \{0,1\}^{N/2} \mid \mathrm{lex}(x) \le 2^{N/2}/2\}$ (where $\mathrm{lex}(x)$ is the lexicographic order of x in $\{0,1\}^{N/2}$) if N is even. Notice that $S^+_{<N/2}$ is a polynomial-time computable set. Then the following algorithm implements W.

Algorithm $Impl_W$**.** Input: $|x\rangle$ such that $\mathrm{wt}(x) < N/2$ in a register S.

1. Create the quantum state $\frac{1}{\sqrt{2}}(|x\rangle + |\overline{x}\rangle)$ in S by Steps 1.1–1.3.
   - 1.1. Prepare $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ in a register R.
   - 1.2. If the content of R is 1, flip all the N bits in S.
   - 1.3. If the content of S is not in $S^+_{<N/2}$, flip the bit in R.

2. Apply $H^{\otimes N}$ on S.
3. Let S be the output.

It is easy to see that $Impl_W$ is implemented in polynomial time. By Step 1.1, we have $\frac{1}{\sqrt{2}}|x\rangle_{\mathsf{S}}(|0\rangle + |1\rangle)_{\mathsf{R}}$. After Step 1.2, the state becomes $\frac{1}{\sqrt{2}}(|x\rangle_{\mathsf{S}}|0\rangle_{\mathsf{R}} + |\bar{x}\rangle_{\mathsf{S}}|1\rangle_{\mathsf{R}})$. Step 1.3 transforms the state to

$$\frac{1}{\sqrt{2}}(|x\rangle_{\mathsf{S}}|0\rangle_{\mathsf{R}} + |\bar{x}\rangle_{\mathsf{S}}|0\rangle_{\mathsf{R}}) = \frac{1}{\sqrt{2}}(|x\rangle_{\mathsf{S}} + |\bar{x}\rangle_{\mathsf{S}})|0\rangle_{\mathsf{R}}.$$

Finally, the state after Step 2 is

$$H^{\otimes N}\left(\frac{1}{\sqrt{2}}(|x\rangle_{\mathsf{S}} + |\bar{x}\rangle_{\mathsf{S}})\right)|0\rangle_{\mathsf{R}} = |\psi_x\rangle_{\mathsf{S}}|0\rangle_{\mathsf{R}}$$

as shown in the proof of Lemma 1.

Appendix B. Algorithm *Find*(k)

The exact algorithm *Find*(k) is given as follows.

Algorithm *Find*(k)**.** Let a ($\geq 9/10$) be the success probability of *Find*$^*(k)$. Let $\mathcal{B}$ be the algorithm that uses a single qubit with initial state $|0\rangle$ and rotates it to $\sqrt{1-1/4a}|0\rangle + \sqrt{1/4a}|1\rangle$. Notice that the probability that *Find*$^*(k)$ succeeds and $\mathcal{B}$ outputs $|1\rangle$ is exactly $1/4$.

(i) Run *Find*$^*(k)$ with initial state $|0\rangle_{\mathsf{R}}$ in the register R and obtain a candidate of k false coins X (in fact, the corresponding oracle), and also run $\mathcal{B}$ with initial state $|0\rangle_{\mathsf{R}'}$ in the register R′. Let U be the unitary transformation done in this step (that is, the state after this step is $U|0\rangle_{\mathsf{R}}|0\rangle_{\mathsf{R}'}$).

(ii) Implement Steps (ii-1)–(ii-3) below.

(ii-1) Run algorithm *Check*, which will be described later, to check if X is indeed the set of k false coins.

(ii-2) If *Check* outputs YES and $\mathcal{B}$ outputs $|1\rangle$, flip the phase. Otherwise, do nothing.

(ii-3) Reverse the operation of Step (ii-1).

(iii) Apply the reflection about the state $U|0\rangle_{\mathsf{R}}|0\rangle_{\mathsf{R}'}$, i.e., $I - 2U|0\rangle\langle 0|U^\dagger$, where $|0\rangle = |0\rangle_{\mathsf{R}}|0\rangle_{\mathsf{R}'}$, to the state.

(iv) Measure R in the computational basis.

By a geometric view (Theorem 4 in [7]) similar to the Grover search where the fraction of correct solution(s) is $1/4$ [6], we can verify that *Find*(k) succeeds with certainty. In *Find*(k), the "solution" is $|X\rangle_{\mathsf{R}}|1\rangle_{\mathsf{R}'}$ where X is the k false coins. Notice that Step (ii) implements the transformation that changes $|X\rangle_{\mathsf{R}}|b\rangle_{\mathsf{R}'}$ to $-|X\rangle_{\mathsf{R}}|b\rangle_{\mathsf{R}'}$ if (X, b) is the "solution" and $|X\rangle_{\mathsf{R}}|b\rangle_{\mathsf{R}'}$ otherwise. The total complexity is the number of queries to run *Find*$^*(k)$ and its inverse three times (once for Step (i) and twice for Step (iii)) plus the number of queries to run *Check* and its inverse. So, we obtain a query complexity of $O(k^{1/4})$ if *Check* has a similar complexity.

In fact, *Check* needs only $O(\log k)$ queries, which is given as follows. For simplicity, we assume that N is a multiple of $k+1$ and $k+1$ is a power of 2 but the generalization is easy. (Note that the following algorithm satisfies the big-pan property. If we do not care the property, the algorithm can be simplified a lot.)

Algorithm *Check*.

Input: Two subsets of a set X of N coins, X_1 with size k and $\overline{X}_1 = X \setminus X_1$ with size $N-k$.

Output: YES iff the coins in X_1 are all false and the coins in $\overline{X}_1$ are all fair.

1. Divide $\overline{X}_1$ into $k+1$ equal-sized subsets $Y_1, Y_2, \ldots, Y_{k+1}$ (recall the above assumption).
2. Let $L = Y_1$ and $R = Y_2$. For $i = 1$ to $\log(k+1)$, repeat Steps 2.1–2.2.
   2.1. Check if L and R are balanced by Steps 2.1.1–2.1.3.
      2.1.1. Construct arbitrarily two subsets L' and R' of size $N/4 - |L|$ ($= N/4 - |R|$) from $X \setminus (X_1 \cup L \cup R)$ (this is possible since $|X \setminus (X_1 \cup L \cup R)| \geq N - k - |L| - |R| \geq (N/4 - |L|) + (N/4 - |R|)$).
      2.1.2. Compare $L \cup L'$ and $R \cup R'$ by a balance scale. If it is tilted, output NO.
      2.1.3. Compare $R \cup L'$ and $L \cup R'$ by a balance scale. If it is tilted, output NO.
   2.2. Set $L := L \cup R$ and $R := \bigcup_{j=2^{i+1}}^{2^{i+1}} Y_j$.
3. Output YES.

Obviously, *Check* makes $O(\log k)$ queries. The correctness of *Check* can be seen as follows: Observe that (i) if L' and R' are of different weight, at least one of Steps 2.1.2 and 2.1.3 is tilted, and (ii) if L' and R' are of the same weight, then both of Steps 2.1.2 and 2.1.3 are balanced if and only if L and R are of the same weight. Hence the algorithm essentially verifies if Y_1 and Y_2 are of the same weight, $Y_1 \cup Y_2$ and $Y_3 \cup Y_4$ are of the same weight, $Y_1 \cup \cdots \cup Y_4$ and $Y_5 \cup \cdots \cup Y_8$ are of the same weight, and so on. If all the tests go through, then Y_1 through Y_{k+1} are all the same weight, which cannot happen if $\overline{X}_1$ includes false coins since $\overline{X}_1$ includes at most k such ones.

Finally, we adapt our algorithm so that it can satisfy the big-pan property. We simulate the transformation $|\widetilde{q}\rangle \mapsto (-1)^{\widetilde{q}\cdot x}|\widetilde{q}\rangle$ of the IP oracle by replacing a query string $\widetilde{q} \in \{0,1\}^N$ with even Hamming weight $\mathrm{wt}(\widetilde{q})$ by two queries

with Hamming weight $\lfloor N/2 \rfloor$ when $\mathrm{wt}(\widetilde{q})/2$ is even (similarly for the case where it is odd). We replace $\widetilde{q}$ by two N-bit strings $\widetilde{q}_1$ and $\widetilde{q}_2$ with Hamming weight $\mathrm{wt}(\widetilde{q})/2$ such that $\widetilde{q} = \widetilde{q}_1 \oplus \widetilde{q}_2$. We take an arbitrary N-bit string $\widetilde{b}$ with $\mathrm{wt}(\widetilde{b}) = \lfloor N/2 \rfloor - \mathrm{wt}(\widetilde{q})/2$ such that $\{i \mid \widetilde{b}_i = 1\} \cap \{i \mid \widetilde{q}_i = 1\} = \emptyset$. Note that both $\widetilde{q}_1 \oplus \widetilde{b}$ and $\widetilde{q}_2 \oplus \widetilde{b}$ have Hamming weight $\lfloor N/2 \rfloor$. (Recall that the Hamming weight of query strings must be even. So, if $\lfloor N/2 \rfloor$ is odd, then we need an adjustment (-1) of the Hamming weight when selecting $\widetilde{b}$.) Since $(-1)^{(\widetilde{q}_1 \oplus \widetilde{b})\cdot x}(-1)^{(\widetilde{q}_2 \oplus \widetilde{b})\cdot x} = (-1)^{\widetilde{q}\cdot x}$ for any x, we can replace a query $\widetilde{q}$ to the IP oracle by two queries $\widetilde{q}_1 \oplus \widetilde{b}$ and $\widetilde{q}_2 \oplus \widetilde{b}$. Thus, we can simulate $\mathit{Find}^*(k)$ without changing the complexity (up to a constant factor).

Appendix C. Other lower bounds for restricted pans

In addition to Theorem 2, we can show more lower bounds for the case where the size of pans is restricted. In what follows, $L \le l$ denotes the restriction that at most l coins must be placed on each pan whenever the balance scale is used.

First, we give a lower bound for the case where the size of pans is "small". Note that Theorem 5 implies that there is no $o(k^{1/4})$-query algorithm placing at most $O(N/\sqrt{k})$ coins on the pans whenever the balance scale is used.

Theorem 5. *If $L \le l$, then any quantum algorithm needs $\Omega(\sqrt{kN/l \min(k, l)})$ queries to find the false k coins from N coins. In particular, $\Omega(\sqrt{N/l})$ queries are necessary.*

Proof. For simplicity, the following weight scheme is given when the size of each pan is l, that is, when query strings are in Q_l. (But the same bound is also obtained similarly when the size is at most l, and hence we can apply Lemma 3 for $Q = \bigcup_{l' \le l} Q_{l'}$ to obtain the desired bound in the last of this proof.) Let $S = \{x \in \{0,1\}^N \mid \mathrm{wt}(x) = k\}$ and $f(x) = x$. For each $(x, y) \in S \times S$, let $w(x, y) = 1$ if $d(x, y) = 2$ and 0 otherwise. When a query q to an oracle x represents that m_1 and m_2 false coins are placed on the left and right pans, respectively, and the query q to another oracle y represents that m_3 and m_4 false coins are placed on the left and right pans, respectively, we put the same weight for all $w'(x, y, q)$'s of such triples (x, y, q), which is denoted as $w'((m_1, m_2), (m_3, m_4))$. Then we define

$$w'((m_1, m_2), (m_3, m_4)) = \begin{cases} \dfrac{m(N-k-(2l-2m))}{(l-m+1)(k-2m+1)} & \text{if } (m_1, m_2, m_3, m_4) = (m-1, m, m, m), (m, m-1, m, m), \\ \dfrac{(l-m+1)(k-2m+1)}{m(N-k-(2l-2m))} & \text{if } (m_1, m_2, m_3, m_4) = (m, m, m-1, m), (m, m, m, m-1), \\ 1 & \text{if one of } m_i\text{'s is } m \text{ and the others are } m-1, \text{ or} \\ & (m_1, m_2, m_3, m_4) = (m+1, m-1, m, m), (m-1, m+1, m, m), \\ & (m, m, m+1, m-1), (m, m, m-1, m+1), \\ 0 & \text{otherwise,} \end{cases}$$

where $1 \le m \le \min(k/2, l)$. It is easy to see that the condition of a weight scheme is satisfied. Notice that for any $x \in S$ we have $\mu(x) = k(N-k)$. Evaluating $\nu(x, q)$ is a bit complicated. Since this value depends on the numbers of false coins on the two pans, m_1 and m_2, represented by the pair (x, q), we denote it by $\nu(m_1, m_2)$. We want to evaluate $\nu(x, q)\nu(y, q)$ such that $w(x, y) > 0$, i.e., $d(x, y) = 2$ and $\chi_x(q) \ne \chi_y(q)$. By symmetry, we can assume that $\chi_x(q) = 1$ and $\chi_y(q) = 0$. Since $d(x, y) = 2$, we need to consider only the following cases: (i) $\nu(x, q) = \nu(m, m-1)$ (or $= \nu(m-1, m)$) and $\nu(y, q) = \nu(m, m)$ (where $0 < m \le \min(k/2, l)$); (ii) $\nu(x, q) = \nu(m+1, m-1)$ (or $= \nu(m-1, m+1)$) and $\nu(y, q) = \nu(m, m)$ (where $0 < m < \min(k/2, l)$); (iii) $\nu(x, q) = \nu(m+1, m)$ (or $= \nu(m, m+1)$) and $\nu(y, q) = \nu(m, m)$ (where $0 \le m < \min(k/2, l)$). In case of (i),

$$\begin{aligned} \nu(x, q) &= \sum_{y: d(x,y)=2,\ \chi_y(q)=0} w'(x, y, q) \\ &= w'((m, m-1), (m-1, m-1)) \times m(N-k-(2l-(2m-1))) \\ &\quad + w'((m, m-1), (m, m)) \times (l-(m-1))(k-(2m-1)) \\ &\le 2m(N-k-2l+2m) \\ &= O(\min(k, l)N), \end{aligned}$$

and

$$\begin{aligned} \nu(y, q) &= \sum_{x: d(x,y)=2,\ \chi_x(q)=1} w'(y, x, q) \\ &= (w'((m, m), (m+1, m)) + w'((m, m), (m, m+1)))(l-m)(k-2m) \\ &\quad + (w'((m, m), (m, m-1)) + w'((m, m), (m-1, m)))m(N-k-(2l-2m)) \\ &\quad + (w'((m, m), (m+1, m-1)) + w'((m, m), (m-1, m+1)))m(l-m) \\ &= 2(l-m)(k-m) + 2(l-m+1)(k-2m+1) \\ &= O(kl), \end{aligned}$$

and hence $\nu(x,q)\nu(y,q) = O(kNl\min(k,l))$. Similarly, in case of (iii), it holds that $\nu(x,q)\nu(y,q) = O(kNl\min(k,l))$. In case of (ii),

$$\begin{aligned}\nu(x,q) &= w'((m+1,m-1),(m,m)) \times (l-(m-1))(m+1)\\ &= (l-m+1)(m+1) = O(\min(k/2,l)l) = O(\min(k,l)N),\end{aligned}$$

and $\nu(y,q) = O(kl)$, and hence we also have $\nu(x,q)\nu(y,q) = O(kNl\min(k,l))$. From the above, by Lemma 3 the quantum query complexity is at least

$$\Omega\left(\min_{\substack{x,y,q:\ w(x,y)>0,\\ \chi_x(q)\neq\chi_y(q)}}\sqrt{\frac{\mu(x)\mu(y)}{\nu(x,q)\nu(y,q)}}\right) = \Omega\left(\sqrt{\frac{kN}{l\min(k,l)}}\right).$$

This completes the proof. □

Second, we generalize Theorem 2 to the case where "big pans" and "small pans" are both available but "medium pans" are not. Here, "$L \le l_1$ or $L \ge l_2$" means that at most l_1 coins or at least l_2 coins (or their superposition) must be placed on each pan whenever the balance scale is used.

Theorem 6. *If $L \le l_1$ or $L \ge l_2$ where $l_1 < l_2$, any quantum algorithm needs $\Omega(\min((N/l_1k)^{1/2}, (l_2k/N)^{1/4}))$ queries to find the k false coins from N coins. In particular, for any $\epsilon \ge 0$, if $L \le N/k^{1+2\epsilon}$ or $L \ge N/k^{1-4\epsilon}$, then $\Omega(k^{\epsilon})$ queries are necessary.*

Proof. We can use the same weight scheme as the proof of Theorem 2. Let $l_1 = N/d_1$ and $l_2 = N/d_2$ with $d_1 > d_2$. The lower bound we should show is $\Omega(\min((d_1/k)^{1/2}, (k/d_2)^{1/4}))$. Similar to the proof of Theorem 2, we can show that by Lemma 3 the quantum query complexity is at least

$$\Omega\left(\min_{\substack{x,y,q\\ w(x,y)>0\\ \chi_x(q)\neq\chi_y(q)}}\sqrt{\frac{\mu(x)\mu(y)}{\nu(x,q)\nu(y,q)}}\right) = \Omega\left(\min_{\substack{c\\ c\ge d_1\\ \text{or } \le d_2}}\sqrt{\frac{\binom{N}{k}}{\gamma(N,k,c)}\cdot\frac{\binom{N}{k}}{\binom{N}{k}-\gamma(N,k,c)}}\right). \tag{C.1}$$

Then the theorem can be obtained from Eq. (C.1) by using Lemma 4 for $c \le d_2$ and the following lemma (Lemma 6) for $c \ge d_1$. (Notice that it suffices to show Lemma 6 for $c \ge 3$ since the bound we should obtain from Lemma 6, $(d_1/k)^{1/2}$, is nontrivial only if $d_1 = \omega(k)$ and hence the size of pans N/c ($\le l_1$) should be considered only for $c = \omega(k)$.)

Lemma 6. $(\binom{N}{k} - \gamma(N,k,c))/\binom{N}{k} = O(\frac{k}{c})$ *for any $c \ge 3$.*

Proof. Let us bound the probability that the balance scale is tilted when N/c coins (N coins include k false ones) are randomly placed on each pan since it is exactly $(\binom{N}{k}-\gamma(N,k,c))/\binom{N}{k}$. Clearly, this probability is upper bounded by the sum $\sum_{m=1}^{k} t'(m)$ where $t'(m) := \frac{\binom{k}{m}\binom{N-k}{2N/c-m}}{\binom{N}{2N/c}}$ denotes the probability of choosing exactly m false coins out of k ones when $2N/c$ coins are placed on the pans. Letting $r'(m) := \frac{t'(m+1)}{t'(m)}$, which is equal to $\frac{(k-m)(2N/c-m)}{(m+1)(N-k-2N/c+m+1)}$, the sum is bounded by

$$\begin{aligned}\sum_{m=1}^{k} t'(m) &\le (r'(0) + r'(0)^2 + \cdots)t'(0) \quad (\text{since } r'(0) \ge r'(m) \text{ for all } m \ge 1)\\ &\le \frac{r'(0)}{1-r'(0)} \quad (\text{by } t'(0) \le 1)\\ &= O(r'(0)).\end{aligned}$$

Since $c \ge 3$ and $k \le N/2$, we can see that the following holds:

$$r'(0) \le \frac{2kN}{cN-ck-2N} = \frac{2k}{c}\cdot\frac{1}{1-\frac{k}{N}-\frac{2}{c}} = O(k/c).$$

This completes the proof. □

Hence the proof of Theorem 6 is completed. □

Unfortunately, Theorem 6 does not give even a weakest nontrivial lower bound $\omega(1)$ if the size of the pans is not restricted. One might have the hope by Theorem 6 that we could obtain a good upper bound by always placing approximately N/k coins on the pans, but Theorem 5 denies such a hope since we have an $\Omega(k^{1/2-2\epsilon})$ lower bound for $l = N/k^{1-4\epsilon}$.

References

[1] A. Ambainis, Quantum lower bounds by quantum arguments, J. Comput. Syst. Sci. 64 (2002) 750–767.
[2] A. Ambainis, Polynomial degree vs. quantum query complexity, J. Comput. Syst. Sci. 72 (2006) 220–238.
[3] A. Ambainis, L. Magnin, M. Roetteler, J. Roland, Symmetry-assisted adversaries for quantum state generation, in: Proc. 26th CCC, 2011, pp. 167–177.
[4] H. Barnum, M.E. Saks, M. Szegedy, Quantum query complexity and semi-definite programming, in: Proc. 18th CCC, 2003, pp. 179–193.
[5] E. Bernstein, U. Vazirani, Quantum complexity theory, SIAM J. Comput. 26 (1997) 1411–1473.
[6] M. Boyer, G. Brassard, P. Høyer, A. Tapp, Tight bounds on quantum searching, Fortschr. Phys. 46 (1998) 493–505.
[7] G. Brassard, P. Høyer, M. Mosca, A. Tapp, Quantum amplitude amplification and estimation, in: Quantum Computation and Quantum Information: A Millennium Volume, in: AMS Contemporary Mathematics Series, vol. 305, 2002, pp. 53–74.
[8] W. van Dam, I. Shparlinski, Classical and quantum algorithms for exponential congruences, Proc. 3rd TQC, Lect. Notes Comput. Sci. 5106 (2008) 1–10.
[9] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proc. 28th STOC, 1996, pp. 212–219.
[10] R.K. Guy, R.J. Nowakowski, Coin-weighing problems, Am. Math. Mon. 102 (1995) 164–167.
[11] L. Halbeisen, N. Hungerbühler, The general counterfeit coin problem, Discrete Math. 147 (1995) 139–150.
[12] P. Høyer, T. Lee, R. Špalek, Negative weights make adversaries stronger, in: Proc. 39th STOC, 2007, pp. 526–535.
[13] S. Laplante, F. Magniez, Lower bounds for randomized and quantum query complexity using Kolmogorov arguments, SIAM J. Comput. 38 (2008) 46–62.
[14] T. Lee, R. Mittal, B.W. Reichardt, R. Špalek, M. Szegedy, Quantum query complexity of state conversion, in: Proc. 52th FOCS, 2011, pp. 344–353.
[15] W.A. Liu, W.G. Zhang, Z.K. Nie, Searching for two counterfeit coins with two-arms balance, Discrete Appl. Math. 152 (2005) 187–212.
[16] F. Magniez, M. Santha, M. Szegedy, Quantum algorithms for the triangle problem, SIAM J. Comput. 37 (2007) 413–424.
[17] B. Manvel, Counterfeit coin problems, Math. Mag. 50 (1977) 90–92.
[18] M. Nielsen, I. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, 2000.
[19] B. Reichardt, Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function, in: Proc. 50th FOCS, 2009, pp. 544–551.
[20] B. Reichardt, Reflections for quantum query algorithms, in: Proc. 22nd SODA, 2011, pp. 560–569.
[21] R. Špalek, M. Szegedy, All quantum adversary methods are equivalent, Theory Comput. 2 (2006) 1–18.
[22] B.M. Terhal, J.A. Smolin, Single quantum querying of a database, Phys. Rev. A 58 (1998) 1822–1826.
[23] S. Zhang, On the power of Ambainis lower bounds, Theor. Comput. Sci. 339 (2005) 241–256.