Quantum Sabotage Complexity

Arjan Cornelissen
Simons Institute for the Theory of Computing, University of California, Berkeley, CA, USA
IRIF – CNRS, Paris, France

Nikhil S. Mande
University of Liverpool, UK

Subhasree Patro
Technische Universiteit Eindhoven, The Netherlands
Centrum Wiskunde en Informatica (QuSoft), Amsterdam, The Netherlands

Abstract

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the goal in the usual query model is to compute f on an unknown input $x \in \{0,1\}^n$ while minimizing the number of queries to x. One can also consider a "distinguishing" problem denoted by f_{sab}: given an input $x \in f^{-1}(0)$ and an input $y \in f^{-1}(1)$, either all differing bits are replaced by a $*$, or all differing bits are replaced by $\dagger$, and an algorithm's goal is to identify which of these is the case while minimizing the number of queries.

Ben-David and Kothari [ToC'18] introduced the notion of randomized sabotage complexity of a Boolean function to be the zero-error randomized query complexity of f_{sab}. A natural follow-up question is to understand the $\mathsf{Q}(f_{\mathsf{sab}})$, the quantum query complexity of f_{sab}. In this paper, we initiate a systematic study of this. The following are our main results for all Boolean functions $f : \{0,1\}^n \to \{0,1\}$.

- If we have additional query access to x and y, then $\mathsf{Q}(f_{\mathsf{sab}}) = O(\min\{\mathsf{Q}(f), \sqrt{n}\})$.
- If an algorithm is also required to output a differing index of a 0-input and a 1-input, then $\mathsf{Q}(f_{\mathsf{sab}}) = O(\min\left\{\mathsf{Q}(f)^{1.5}, \sqrt{n}\right\})$.
- $\mathsf{Q}(f_{\mathsf{sab}}) = \Omega(\sqrt{\mathsf{fbs}(f)})$, where $\mathsf{fbs}(f)$ denotes the fractional block sensitivity of f. By known results, along with the results in the previous bullets, this implies that $\mathsf{Q}(f_{\mathsf{sab}})$ is polynomially related to $\mathsf{Q}(f)$.
- The bound above is easily seen to be tight for standard functions such as And, Or, Majority and Parity. We show that when f is the Indexing function, $\mathsf{Q}(f_{\mathsf{sab}}) = \Theta(\mathsf{fbs}(f))$, ruling out the possibility that $\mathsf{Q}(f_{\mathsf{sab}}) = \Theta(\sqrt{\mathsf{fbs}(f)})$ for all f.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees; Theory of computation → Quantum complexity theory

Keywords and phrases Sabotage complexity, quantum query complexity, Boolean functions, fractional block sensitivity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2024.19

Related Version *Full Version*: `https://arxiv.org/abs/2408.12595`

Funding *Subhasree Patro*: This work is co-funded by the European Union (ERC, ASC-Q, 101040624).

Acknowledgements We thank the anonymous reviewers for useful comments and pointers to related results that we had missed.

1 Introduction

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the goal in the standard query complexity model is to compute $f(x)$ on an unknown input $x \in \{0,1\}^n$ using as few queries to x as possible. One can also consider the following distinguishing problem: given $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$, output an index $i \in [n]$ such that $x_i \neq y_i$. This task can be formulated as follows: Consider an arbitrary $x \in f^{-1}(0)$, an arbitrary $y \in f^{-1}(1)$, and then either all indices where

x and y differ are replaced by the symbol $*$, or all such indices are replaced by the symbol $\dagger$. The goal of an algorithm is to identify which of these is the case, with query access to this "sabotaged" input. A formal description of this task is given below.

Let $f : D \to \{0,1\}$ with $D \subseteq \{0,1\}^n$ be a (partial) Boolean function. Let $D_0 = f^{-1}(0)$ and $D_1 = f^{-1}(1)$. For any pair $(x,y) \in D_0 \times D_1$, define $[x,y,*] \in \{0,1,*\}^n$ to be

$$[x,y,*]_i = \begin{cases} x_i, & \text{if } x_i = y_i, \\ *, & \text{otherwise.} \end{cases}$$

Similarly, for any pair $(x,y) \in D_0 \times D_1$, define $[x,y,\dagger] \in \{0,1,\dagger\}^n$ to be

$$[x,y,\dagger]_i = \begin{cases} x_i, & \text{if } x_i = y_i, \\ \dagger, & \text{otherwise.} \end{cases}$$

Let $S_* = \{[x,y,*] \mid (x,y) \in D_0 \times D_1\}$ and $S_\dagger = \{[x,y,\dagger] \mid (x,y) \in D_0 \times D_1\}$. That is, S_* is the set of $*$-sabotaged inputs for f and $S_\dagger$ is the set of $\dagger$-sabotaged inputs for f. Finally, let $f_{\mathsf{sab}} : S_* \cup S_\dagger \to \{0,1\}$ be the function defined by

$$f_{\mathsf{sab}}(z) = \begin{cases} 0, & z \in S_*, \\ 1, & z \in S_\dagger. \end{cases}$$

That is, f_{sab} takes as input a sabotaged input to f and identifies if the input is $*$-sabotaged or if it is $\dagger$-sabotaged.

Ben-David and Kothari [10] introduced the notion of *randomized sabotage complexity* of a Boolean function f, defined to be $\mathsf{RS}(f) := \mathsf{R}_0(f_{\mathsf{sab}})$, where $\mathsf{R}_0(\cdot)$ denotes randomized zero-error query complexity. It is not hard to see that $\mathsf{RS}(f) = O(\mathsf{R}(f))$ (where $\mathsf{R}(\cdot)$ denotes randomized bounded-error query complexity); this is because a randomized algorithm that succeeds with high probability on both a 0-input x and a 1-input y must, with high probability, query an index where x and y differ. Ben-David and Kothari also showed that randomized sabotage complexity admits nice composition properties. It is still open whether $\mathsf{RS}(f) = \Theta(\mathsf{R}(f))$ for all total Boolean functions f. If true, this would imply that randomized query complexity admits a perfect composition theorem, a goal towards which a lot of research has been done [5, 16, 14, 8, 6, 9, 13, 24]. This motivates the study of randomized sabotage complexity.

In the same paper, they mentioned that one could define $\mathsf{QS}(f) := \mathsf{Q}(f_{\mathsf{sab}})$ (here, $\mathsf{Q}(\cdot)$ denotes bounded-error quantum query complexity), but they were unable to show that it lower bounds $\mathsf{Q}(f)$. In a subsequent work [11], they defined a quantum analog, denoted $\mathsf{QD}(f)$, and called it *quantum distinguishing complexity.* $\mathsf{QD}(f)$ is the minimum number of queries to the input $x \in D$ to produce an output state such that the output states corresponding to 0-inputs and 1-inputs are far from each other. Analogous to their earlier result, they were also able to show that $\mathsf{QD}(f) = O(\mathsf{Q}(f))$ for all total f. Additionally, using $\mathsf{QD}(f)$ as an intermediate measure, they were able to show a (then) state-of-the-art 5th-power relationship between zero-error quantum query complexity and bounded-error quantum query complexity: $\mathsf{Q}_0(f) = \widetilde{O}(\mathsf{Q}(f)^5)$ for all total f. We note here that a 4th-power relationship was subsequently shown between $\mathsf{D}(f)$ and $\mathsf{Q}(f)$ [2], also implying $\mathsf{Q}_0(f) = O(\mathsf{Q}(f)^4)$ for all total f. The proof of this relied on Huang's celebrated sensitivity theorem [19].

Our results

In this paper, we initiate a systematic study of the natural quantum analog of randomized sabotage complexity alluded to in the previous paragraph, which we call *quantum sabotage complexity*, denoted by $\mathsf{QS}(f) := \mathsf{Q}(f_{\mathsf{sab}})$. Slightly more formally, we consider the following variants:

- We consider two input models. In the weak input model, the oracle simply has query access to an input in $z \in S_* \cup S_\dagger$. In the strong input model, the oracle additionally has access to the original inputs $x \in D_0$ and $y \in D_1$ that yield the corresponding input in $S_* \cup S_\dagger$. The model under consideration will be clear by adding either weak or str as a subscript to QS.
- We also consider two different output models: one where an algorithm is only required to output whether the input was in S_* or in $S_\dagger$, and a stronger version where an algorithm is required to output an index $i \in [n]$ with $z_i \in \{*, \dagger\}$. The model under consideration will be clear by adding either no superscript or ind as a superscript to QS.

As an example, $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{weak}}(f)$ denotes the quantum sabotage complexity of f under the weak input model, and in the output model where an algorithm needs to output a differing index.

One can also consider these nuances in defining the input and output models in the randomized setting. However, one can easily show that they are all equivalent for randomized algorithms. We refer the reader to Section 3 for a proof of this, and for a formal description of these models.

An immediate upper bound on $\mathsf{QS}(f)$, for any (partial) Boolean function f, follows directly from Grover's search algorithm [17]. Indeed, for any sabotaged input, we know that at least one of the input symbols must be either a $*$ or $\dagger$. Thus, we can simply use the unstructured search algorithm by Grover to find (the position of) such an element in $O(\sqrt{n})$ queries. This immediately tells us that for Boolean functions where $\mathsf{Q}(f) = \omega(\sqrt{n})$, $\mathsf{QS}(f)$ is significantly smaller than $\mathsf{Q}(f)$.

As mentioned earlier, Ben-David and Kothari left open the question of whether $\mathsf{QS}(f) = O(\mathsf{Q}(f))$, the quantum analog of randomized sabotage complexity being at most randomized query complexity. We first observe that in the strong input model, this holds true.

▶ **Lemma 1.** *Let n be a positive integer, let $D \subseteq \{0,1\}^n$. Let $f : D \to \{0,1\}$ be a (partial) Boolean function. Then, $\mathsf{QS}_{\mathsf{str}}(f) = O(\mathsf{Q}(f))$.*

The proof idea is simple: consider an input $z \in S^* \cup S^\dagger$ obtained by sabotaging $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$. Run a quantum query algorithm for f on input z, such that:

- Whenever a bit in $\{0,1\}$ is encountered, the algorithm proceeds as normal.
- Whenever a $*$ is encountered, query the corresponding bit in x.
- Whenever a $\dagger$ is encountered, query the corresponding bit in y.

The correctness follows from the following observation: if $z \in S^*$, then the run of the algorithm is exactly that of the original algorithm on x, and if $z \in S^\dagger$, then the run is the same of the original algorithm on y.

This procedure does not return a $*/\dagger$-index, and it is natural to ask if $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{str}}(f) = O(\mathsf{Q}(f))$ as well. While we are unable to show this, we make progress towards this by showing the following, which is our first main result.

▶ **Theorem 2.** *Let n be a positive integer, let $D \subseteq \{0,1\}^n$, and let $f : D \to \{0,1\}$ be a (partial) Boolean function. Then, $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{str}}(f) = O(\mathsf{Q}(f)^{1.5})$.*

Our result is actually slightly stronger than this; our proof shows that $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{str}}(f) = O(\mathsf{QD}(f)^{1.5})$. This implies Theorem 2 using the observation of Ben-David and Kothari that $\mathsf{QD}(f) = O(\mathsf{Q}(f))$ for all (partial) f.

In order to show Theorem 2, we take inspiration from the observation that randomized sabotage complexity is at most randomized query complexity. This is true because with high probability, a randomized query algorithm must spot a differing bit between any pair of inputs with different output values. However, this argument cannot immediately be ported to the quantum setting because quantum algorithms can make queries in superposition. We are able to do this, though, by stopping a quantum query algorithm for f at a random time and measuring the index register. We note here that it is important that we have oracle access not only to a sabotaged input $z \in S_* \cup S_\dagger$, but also the inputs $(x, y) \in D_0 \times D_1$ that yielded the underlying sabotaged input. Using arguments reminiscent of the arguments in the hybrid method [12], we are able to show that the success probability of this is only $1/\mathsf{Q}(f)$. Applying amplitude amplification to this process leads to an overhead of $\sqrt{\mathsf{Q}(f)}$, and yields Theorem 2.

We remark here that this proof idea is reminiscent of the proof of [11, Theorem 1]. However, there are some technical subtleties. At a high level, the main subtleties are the following: in the strong oracle model that we consider, it is easy to check whether or not a terminated run of a Q algorithm actually gives us a $*/\dagger$ index. This is not the case in [11, Proof of Theorem 1]. This allows us to save upon a quadratic factor because we can do amplitude amplification.

Our proof approach modifies a core observation by Ben-David and Kothari [11, Lemma 12]. In their setting, they consider a T-query algorithm $\mathcal{A}$ that computes a function f with high probability. Take two inputs x and y such that $f(x) \neq f(y)$, and let $B \subseteq [n]$ be the indices on which x and y differ. Suppose we run this algorithm on x, interrupt it right before the tth query, and then measure the query register. The probability that we measure an index $i \in B$, we denote by p_t. Then, [11, Lemma 12] shows that

$$\sum_{t=1}^{T} p_t = \Omega\left(\frac{1}{T}\right).$$

We show that by adding the probabilities that come from running the algorithm at input y, we can replace the lower bound of $\Omega(1/T)$ by a much stronger lower bound of $\Omega(1)$. We state this observation more formally in the following lemma.

▶ **Lemma 3.** *Let $x \in \{0,1\}^n$ be an input and let $B \subseteq [n]$. Let $\mathcal{A}$ be a T-query quantum algorithm that accepts x and rejects x_B with high probability, or more generally produces output states that are a constant distance apart in trace distance for x and x_B. Let p_t, resp. p_t^B, be the probability that, when $\mathcal{A}$ is run on x, resp. x_B, up until, but not including, the t-th query and then measured, it is found to be querying a position $i \in B$. Then,*

$$\sum_{t=1}^{T} (p_t + p_t^B) = \Omega(1).$$

▶ Remark 4. We remark here that a stronger statement than that in Lemma 3 has already been shown in [21, Lemma 3.1]. We thank an anonymous reviewer for pointing this out to us.

We find this lemma independently interesting and are confident that it will find use in future research, given the use of such statements in showing quantum lower bounds via the adversary method, for example (see [26, Chapters 11-12] and the references therein). Note that this lemma is very amenable to our “strong” sabotage complexity setup: imagine running an algorithm simultaneously on inputs x and x_B that have different function values. Lemma 3 says that on stopping at a random time in the algorithm, and choosing one of x and x_B at random, the probability of seeing an index in B (i.e., a $*$-index or a $\dagger$-index) is a constant. Indeed, this lemma is a natural quantum generalization of the phenomenon that occurs in the randomized setting: a randomized algorithm that distinguishes x and x_B must read an index in B with constant probability (on input either x or x_B).

We now discuss our remaining results. Given Theorem 2, it is natural to ask if $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(\cdot)$ is polynomially related to $\mathsf{Q}(\cdot)$. Using the positive-weighted adversary lower bound for quantum query complexity [4], we are able to show that $\mathsf{QS}_{\mathsf{str}}(f) = \Omega(\sqrt{\mathsf{fbs}(f)})$ for all total f (and hence the same lower bound holds for $\mathsf{QS}_{\mathsf{weak}}(f)$, and $\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f)$, and $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f)$ as well).

▶ **Theorem 5.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then $\mathsf{QS}_{\mathsf{str}}(f) = \Omega(\sqrt{\mathsf{fbs}(f)})$.*

Fractional block sensitivity is further lower bounded by block sensitivity, which is known to be polynomially related to $\mathsf{R}(\cdot)$ and $\mathsf{Q}(\cdot)$ [23, 7]. Using the best-known relationship of $\mathsf{Q}(f) = \widetilde{O}(\mathsf{bs}^3(f))$ [1], Theorem 2 and Theorem 5 implies the following polynomial relationship between $\mathsf{QS}_{\mathsf{str}}(f)$, $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f)$ and $\mathsf{Q}(f)$ for all total Boolean functions f.

$$\mathsf{QS}_{\mathsf{str}}(f) = O(\mathsf{Q}(f)), \qquad \mathsf{Q}(f) = \widetilde{O}(\mathsf{QS}_{\mathsf{str}}(f)^6). \tag{1}$$

$$\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f) = O(\mathsf{Q}(f)^{1.5}), \qquad \mathsf{Q}(f) = \widetilde{O}(\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f)^6). \tag{2}$$

In the weakest input model, we have $\mathsf{QS}_{\mathsf{weak}}(f) = O(\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f)) = O(\mathsf{R}(f_{\mathsf{sab}})) = O(\mathsf{R}(f)) = O(\mathsf{Q}(f)^4)$ (where the last inequality follows from [2]). Thus, in the weakest input model for sabotage complexity, Theorem 5 implies the following polynomial relationship with $\mathsf{Q}(f)$:

$$\mathsf{QS}_{\mathsf{weak}}(f) = O(\mathsf{Q}(f)^4), \qquad \mathsf{Q}(f) = \widetilde{O}(\mathsf{QS}_{\mathsf{weak}}(f)^6). \tag{3}$$

$$\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f) = O(\mathsf{Q}(f)^4), \qquad \mathsf{Q}(f) = \widetilde{O}(\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f)^6). \tag{4}$$

It would be interesting to find the correct polynomial relationships between all of these measures. In particular, we suspect that $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f) = O(\mathsf{Q}(f))$ for all total f, but we are unable to show this.

As mentioned in the discussion before Theorem 2, it is easy to show that $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f) = O(\sqrt{n})$ for all $f : \{0,1\}^n \to \{0,1\}$. Thus, the lower bound in terms of block sensitivity given by Theorem 5 is actually tight for standard functions like And, Or, Parity and Majority. Given this, it is natural to ask if $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f) = \Theta(\sqrt{\mathsf{fbs}(f)})$ for all total Boolean f. We show that this is false, by showing that for the Indexing function $\mathsf{IND}_n : \{0,1\}^{n+2^n} \to \{0,1\}$ defined by $\mathsf{IND}_n(x,y) = y_{\mathsf{bin}(x)}$ (where $\mathsf{bin}(x)$ denotes the integer in $[2^n]$ with the binary representation x), $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(\mathsf{IND}_n) = \Theta(\mathsf{fbs}(\mathsf{IND}_n)) = \Theta(n)$.

▶ **Theorem 6.** *Let n be a positive integer. Then, $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(\mathsf{IND}_n) = \Theta(n)$.*

In order to show this, we use a variation of Ambainis' basic adversary method [3, Theorem 5.1], also presented in the same paper [3, Theorem 6.1] (see Lemma 19).

Finally, we summarize the relations we proved in Figure 1.

2 Preliminaries

All logarithms in this paper are taken base 2 unless mentioned otherwise. For a positive integer n, we use the notation $[n]$ to denote the set $\{1, 2, \ldots, n\}$ and use $[n]_0$ to denote $\{0, 1, \ldots, n-1\}$. Let $v \in \mathbb{C}^d$, the $\|v\|_2 = \sqrt{\sum_{i=1}^d |v_i|^2}$. Let A, B be square matrices in $\mathbb{C}^{d\times d}$. We use $A^\dagger$ to denote the conjugate transpose of matrix A. The operator norm of a matrix A, denoted by $\|A\|$, is the largest singular value of A, i.e., $\|A\| = \max_{v:\|v\|_2=1} \|Av\|_2$. The trace distance between two matrices A, B, denoted by $\|A - B\|_{\mathsf{tr}} = \frac{1}{2}\|A - B\|_1$ where $\|A\|_1 = \mathrm{Tr}(\sqrt{A^\dagger A})$. For two $d \times d$ matrices A, B, $A \circ B$ denotes the Hadamard, or entry-wise, product of A and B.

We refer the reader to [26, Chapter 1] for the relevant basics of quantum computing.

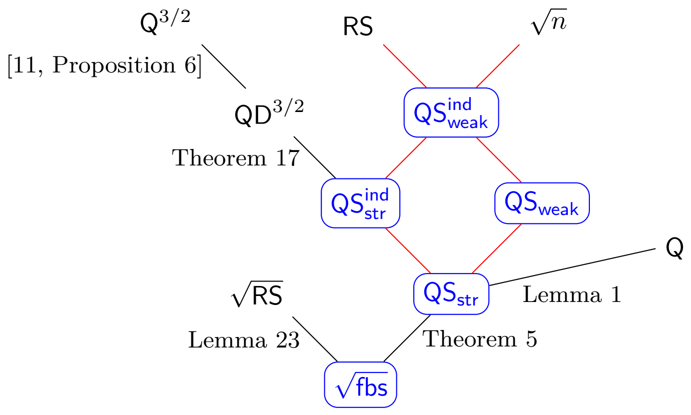

Figure 1 Overview of the relations proved in this work. If nodes A and B are connected, and A is below B, then $A = O(B)$. All the red edges are reasonably straightforward inclusions, and they are proved in Proposition 16.

Let $D \subseteq \{0,1\}^n$, let R be a finite set, and let $f \subseteq D \times R$ be a relation. A quantum query algorithm $\mathcal{A}$ for f begins in a fixed initial state $|\psi_0\rangle$ in a finite-dimensional Hilbert space, applies a sequence of unitaries $U_0, O_x, U_1, O_x, \ldots, U_T$, and performs a measurement. Here, the initial state $|\psi_0\rangle$ and the unitaries $U_0, U_1, \ldots, U_T$ are independent of the input. The unitary O_x represents the "query" operation, and does the following for each basis state: it maps $|i\rangle |b\rangle$ to $|i\rangle |b + x_i \mod 2\rangle$ for all $i \in [n]$. The algorithm then performs a measurement and outputs the observed value. We say that $\mathcal{A}$ is a bounded-error quantum query algorithm computing f if for all $x \in D$ the probability of outputting r such that $(x,r) \in f$ is at least $2/3$. The (bounded-error) *quantum query complexity of* f, denoted by $\mathsf{Q}(f)$, is the least number of queries required for a quantum query algorithm to compute f with error at most $1/3$. We use $\mathsf{R}(f)$ to denote the *randomized query complexity* of f, which is the worst-case cost (number of queries) of the best randomized algorithm that computes f to error at most $1/3$ on all inputs.

We recall some known complexity measures.

▶ **Definition 7** (Block sensitivity). *Let n be a positive integer and $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. For any $x \in \{0,1\}^n$, a block $B \subseteq [n]$ is said to be sensitive on an input $x \in \{0,1\}^n$ if $f(x) \neq f(x \oplus B)$, where $x \oplus B$ (or x_B) denotes the string obtained by taking x and flipping all bits in B. The* block sensitivity of f on x, *denoted* $\mathsf{bs}(f,x)$, *is the maximum number of pairwise disjoint blocks that are sensitive on x. The* block sensitivity of f, *denoted by* $\mathsf{bs}(f)$, *is* $\max_{x \in \{0,1\}^n} \mathsf{bs}(f,x)$.

The *fractional block sensitivity* of f on x, denoted $\mathsf{fbs}(f,x)$, is the optimum value of the linear program below. We refer the reader to [15, 20] for a formal treatment of fractional block sensitivity and related measures, and we simply state its definition here.

▶ **Definition 8** (Fractional block sensitivity). *Let n be a positive integer and $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Let $x \in \{0,1\}^n$ and let $Y_x = \{z \in \{0,1\}^n : f(x) \neq f(z)\}$. The fractional block sensitivity of f on x, denoted by* $\mathsf{fbs}(f,x)$, *is the optimal value of the following optimization program.*

$$\begin{aligned} \max \quad & \sum_{y \in Y_x} w_y, & \\ \text{subject to} \quad & \sum_{\substack{y \in Y_x \\ x_j \neq y_j}} w_y \leq 1, & \forall j \in [n], \\ & w_y \geq 0, & \forall y \in Y_x. \end{aligned}$$

The fractional block sensitivity of f, denoted by $\mathsf{fbs}(f)$, *is* $\max_{x \in \{0,1\}^n} \mathsf{fbs}(f,x)$.[1]

We also state the non-negative weight adversary bound. It appears in many forms in the literature. We refer to the form mentioned in [18] which is equivalent to the following definition.

▶ **Definition 9** (Non-negative weight adversary bound). *Let n be a positive integer, Σ and Π finite sets, $D \subseteq (\Sigma)^n$, and $f : D \to \Pi$. The adversary bound is the following optimization program.*

$$\begin{aligned} \max \quad & \|\Gamma\| & \\ s.t. \quad & \|\Gamma \circ \Delta_j\| \leq 1, & \forall j \in [n], \\ & \Gamma[x,y] = 0, & \text{if } f(x) = f(y). \end{aligned}$$

Here, the optimization is over all symmetric, entry-wise non-negative adversary matrices $\Gamma \in \mathbb{R}^{D \times D}$. The matrix $\Delta_j \in \{0,1\}^{D \times D}$ has entries $\Delta_j[x,y] = 1$ if and only if $x_j \neq y_j$. The optimal value of this optimization program is denoted by $\mathsf{ADV}^+(f)$.

Sometimes, the optimal value of the non-negative weight adversary bound is also written as $\mathsf{ADV}(f)$. However, one can also consider the general adversary bound, in which the entries of the matrix Γ are not constrained to be non-negative. To clearly differentiate between the optimal values of these optimization programs, we distinguish between them by explicitly writing $\mathsf{ADV}^+(f)$ and $\mathsf{ADV}^{\pm}(f)$.

▶ **Definition 10** (Quantum distinguishing complexity). *Let n be a positive integer. The quantum distinguishing complexity of a (partial) Boolean function $f : D \to \{0,1\}$ (where $D \subseteq \{0,1\}^n$), denoted by* $\mathsf{QD}(f)$, *is the smallest integer k such that there exists a k-query algorithm that on input $x \in D$ outputs a quantum state ρ_x such that*

$$\forall x, y \in D, \quad \|\rho_x - \rho_y\|_{\mathsf{tr}} \geq 1/6,$$

whenever $f(x) \neq f(y)$.

The non-negative weight adversary bound is known to be a lower bound to the quantum distinguishing complexity, which in turn is a lower bound on the quantum query complexity, by [11, Proposition 6]. We state it below.

▶ **Theorem 11** ([11]). *Let n be a positive integer, $D \subseteq \{0,1\}^n$, and $f : D \to \{0,1\}$. Then,*

$$\mathsf{ADV}^+(f) = O(\mathsf{QD}(f)) = O(\mathsf{Q}(f)).$$

The relation $\mathsf{ADV}^+(f) = O(\mathsf{Q}(f))$ holds in the non-Boolean case as well, which follows directly from the definition and known results about the non-negative weight adversary bound, as can be found in [22], for instance.

[1] The block sensitivity of f on x is captured by the integral version of this linear program, where the variable $w_y \in \{0,1\}$ enforces that the blocks must be disjoint.

3 Sabotage complexity

In this section we first define sabotage variants of a Boolean function f that are convenient to work with. Specifically, these variants are useful because they enable us to work with the usual quantum query complexity model in the quantum setting, allowing us to use known results in this setting. After this, we analyze some basic properties of quantum sabotage complexities.

3.1 Formal setup of sabotage complexity

We start by formally defining the sabotage function of f.

▶ **Definition 12** (Sabotage functions and relations). *Let n be a positive integer, $D \subseteq \{0,1\}^n$. Let $f : D \to \{0,1\}$ be a (partial) Boolean function. For any input pair $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$, we define $[x,y,*], [x,y,\dagger] \in \{0,1,*,\dagger\}^n$ by*

$$[x,y,*]_j = \begin{cases} x_j, & \text{if } x_j = y_j, \\ *, & \text{otherwise,} \end{cases} \qquad \text{and} \qquad [x,y,\dagger]_j = \begin{cases} x_j, & \text{if } x_j = y_j, \\ \dagger, & \text{otherwise.} \end{cases}$$

We let $S_* = \{[x,y,*] : x \in f^{-1}(0), y \in f^{-1}(1)\}$, $S_\dagger = \{[x,y,\dagger] : x \in f^{-1}(0), y \in f^{-1}(1)\}$, and we let $D_{\mathsf{sab}} = S_* \cup S_\dagger \subseteq \{0,1,*,\dagger\}^n$.

- *The* sabotage function of f *is defined as $f_{\mathsf{sab}} : D_{\mathsf{sab}} \to \{0,1\}$, where $f_{\mathsf{sab}}(z) = 1$ iff $z \in S_\dagger$.*
- *The* sabotage relation of f *is defined as $f_{\mathsf{sab}}^{\mathsf{ind}} \subseteq D_{\mathsf{sab}} \times [n]$, where $(z,j) \in f_{\mathsf{sab}}^{\mathsf{ind}}$ iff $z_j \in \{*,\dagger\}$.*

For every $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$, we let $(x,y,*)$ denote $((x_j,y_j,z_j))_{j=1}^n$, where $z = [x,y,*]$. Similarly, for every $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$, we let $(x,y,\dagger)$ denote $((x_j,y_j,z_j))_{j=1}^n$, where $z = [x,y,\dagger]$. For $b \in \{*,\dagger\}$, let $S_b^{\mathsf{str}} = \{(x,y,b) : x \in f^{-1}(0), y \in f^{-1}(1)\}$, and $D_{\mathsf{sab}}^{\mathsf{str}} = S_*^{\mathsf{str}} \cup S_\dagger^{\mathsf{str}}$.

- *We define the* strong sabotage function of f *as $f_{\mathsf{sab}}^{\mathsf{str}} : D_{\mathsf{sab}}^{\mathsf{str}} \to \{0,1\}$, where $f_{\mathsf{sab}}^{\mathsf{str}}(w) = 1$ iff $w \in S_\dagger^{\mathsf{str}}$.*
- *We define the* strong sabotage relation of f *as $f_{\mathsf{sab}}^{\mathsf{str,ind}} \subseteq D_{\mathsf{sab}}^{\mathsf{str}} \times [n]$, where $(w,j) \in f_{\mathsf{sab}}^{\mathsf{str,ind}}$ iff $z_j \in \{*,\dagger\}$ where $w = ((x_j,y_j,z_j))_{j=1}^n$.*

If we want to compute f_{sab}, we need to consider how we are given access to the input of f_{sab}. To that end, we consider two input models, the weak and the strong input model. Both can be viewed as having regular query access to the inputs of the function f_{sab} and $f_{\mathsf{sab}}^{\mathsf{str}}$, respectively.

▶ **Definition 13** (Weak sabotage input model). *Let n be a positive integer, $D \subseteq \{0,1\}^n$, and $f : D \to \{0,1\}$ be a (partial) Boolean function. Let D_{sab} be as in Definition 12. In the* weak sabotage input model*, on some input $z \in D_{\mathsf{sab}}$, we are given access to an oracle O_z^{weak} that when queried the jth position returns z_j. In the quantum setting, this means that the oracle performs the mapping*

$$O_z^{\mathsf{weak}} : |j\rangle\,|b\rangle \mapsto |j\rangle\,|(b + z_j) \mod 4\rangle\,, \quad \forall b \in [4]_0, \forall j \in [n],$$

where $*$ is identified with 2 and $\dagger$ is identified with 3.

We also consider a stronger model.

▶ **Definition 14** (Strong sabotage input model). *Let n be a positive integer, $D \subseteq \{0,1\}^n$, and $f : D \to \{0,1\}$ be a (partial) Boolean function. Let $D_{\mathsf{sab}}^{\mathsf{str}}$ be as in Definition 12. In the* strong sabotage input model*, on some input $w = ((x_j, y_j, z_j))_{j=1}^n$, we are given access to an oracle O_w^{str} that when queried the jth position returns the tuple (x_j, y_j, z_j). In the quantum setting, this means that the oracle performs the mapping*

$$O_w^{\mathsf{str}} : |j\rangle |b_x\rangle |b_y\rangle |b_z\rangle \mapsto |j\rangle |b_x \oplus x_j\rangle |b_y \oplus y_j\rangle |(b_z + z_j) \mod 4\rangle ,$$

for all $b_x, b_y \in \{0,1\}$, $b_z \in [4]_0$ and for all $j \in [n]$. As in the previous definition, $*$ is identified with 2 and $\dagger$ is identified with 3.

Note that in the stronger model, we are implicitly also given the information which of the two inputs x and y are the 0- and 1-inputs of f. Indeed, we assume that the bits queried in the first entry of the tuple, always correspond to the 0-input that defined the sabotaged input z. We remark here that we can always remove this assumption if we allow for additive overhead of $O(\mathsf{Q}(f))$, after all we can always run a quantum algorithm that computes f on the first bits from all our queried tuple, to compute $f(x)$, and thus finding out whether it was a 0- or 1-input to begin with.

Having defined the sabotage functions/relations and the input models we now define four different notions of quantum sabotage complexity of f.

▶ **Definition 15** (Sabotage complexity). *Let n be a positive integer, $D \subseteq \{0,1\}^n$, and let $f : D \to \{0,1\}$ be a (partial) Boolean function. Define,*

$$\mathsf{QS}_{\mathsf{weak}}(f) := \mathsf{Q}(f_{\mathsf{sab}}), \quad \mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f) := \mathsf{Q}(f_{\mathsf{sab}}^{\mathsf{ind}}), \quad \mathsf{QS}_{\mathsf{str}}(f) := \mathsf{Q}(f_{\mathsf{sab}}^{\mathsf{str}}), \quad \mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f) := \mathsf{Q}(f_{\mathsf{sab}}^{\mathsf{str,ind}}).$$

Analogously, define

$$\mathsf{RS}_{\mathsf{weak}}(f) := \mathsf{R}(f_{\mathsf{sab}}), \quad \mathsf{RS}_{\mathsf{weak}}^{\mathsf{ind}}(f) := \mathsf{R}(f_{\mathsf{sab}}^{\mathsf{ind}}), \quad \mathsf{RS}_{\mathsf{str}}(f) := \mathsf{R}(f_{\mathsf{sab}}^{\mathsf{str}}), \quad \mathsf{RS}_{\mathsf{str}}^{\mathsf{ind}}(f) := \mathsf{R}(f_{\mathsf{sab}}^{\mathsf{str,ind}}).$$

3.2 Quantum sabotage complexity

In Appendix A we show that the randomized sabotage complexity of a function is (asymptotically) the same in all of the four models we consider. In the quantum case, we have not been able to prove such equivalences. However, we can still prove some bounds between them. We refer the reader to Figure 1 for a pictorial representation of all relationships.

▶ **Proposition 16.** *Let n be a positive integer, $D \subseteq \{0,1\}^n$, and let $f : D \to \{0,1\}$ be a (partial) Boolean function. Then,*

$$\mathsf{QS}_{\mathsf{str}}(f) = O(\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f)) = O(\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f)) = O(\min\{\mathsf{RS}(f)), \sqrt{n}\}),$$

and

$$\mathsf{QS}_{\mathsf{str}}(f) = O(\mathsf{QS}_{\mathsf{weak}}(f)) = O(\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f)).$$

Proof. Just as in the randomized case, $\mathsf{QS}_{\mathsf{str}}(f) = O(\mathsf{QS}_{\mathsf{weak}}(f))$ and $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f) = O(\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}})(f)$, because the input model is stronger, i.e., we can simulate a query to the weak oracle with $O(1)$ queries to the strong oracle. Furthermore, $\mathsf{QS}_{\mathsf{str}}(f) = O(\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f))$ and $\mathsf{QS}_{\mathsf{weak}}(f) = O(\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f))$, because once we have found a $j \in [n]$ where $x_j \neq y_j$, we can query that index with one more query to find figure out whether we have a $*$-input or a $\dagger$-input.

In the strong input model, we can always simply run Grover's algorithm to find a position $j \in [n]$ where $z_j \in \{*, \dagger\}$. This takes $O(\sqrt{n})$ queries. Thus, it remains to show that $\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f) = O(\mathsf{RS}(f))$. We showed in the previous proposition that $\mathsf{RS}(f)$ is the same up to constants to $\mathsf{RS}_{\mathsf{weak}}^{\mathsf{ind}}(f)$, and since the quantum computational model is only stronger than the randomized one, we find $\mathsf{QS}_{\mathsf{weak}}^{\mathsf{ind}}(f) = O(\mathsf{RS}_{\mathsf{weak}}^{\mathsf{ind}}(f)) = O(\mathsf{RS}(f))$. ◀

4 Upper bounds on QS

In this section, we prove upper bounds on the complexity measures introduced in Section 3. We start by showing that the quantum sabotage complexity in the strong model is upper bounded by the regular bounded-error quantum query complexity. Thereby, we prove that $\mathsf{QS}_{\mathsf{str}}(f)$ has the property that was sought for in [10], i.e., in this model computing f_{sab} indeed costs at most as many queries as computing f itself.

▶ **Lemma 1.** *Let n be a positive integer, let $D \subseteq \{0,1\}^n$. Let $f : D \to \{0,1\}$ be a (partial) Boolean function. Then, $\mathsf{QS}_{\mathsf{str}}(f) = O(\mathsf{Q}(f))$.*

Proof. Let $\mathcal{A}$ be a bounded-error quantum query algorithm that computes f. We construct a quantum query algorithm $\mathcal{B}$ that computes f_{sab} in the strong input model.

Recall that in the strong input model (as in Definition 14), our input is viewed as $w = ((x_j, y_j, z_j))_{j=1}^n$ where $f(x) = 0, f(y) = 1$ and z is the sabotaged input constructed from x and y. A query on the jth position to the oracle O_w^{str} returns a tuple (x_j, y_j, z_j). Now, we define $\mathcal{B}$ to be the same algorithm as $\mathcal{A}$, but whenever $\mathcal{A}$ makes a query, it performs the following operation instead:

1. Query O_w^{str}, denote the outcome by (x_j, y_j, z_j).
2. If $z_j \in \{0,1\}$, return z_j.
3. If $z_j = *$, return x_j.
4. If $z_j = \dagger$, return y_j.

Note that this operation can indeed be implemented quantumly making 2 queries to O_w^{str}. The initial query performs the instructions described above, and the second query uncomputes the values from the tuple we don't need for the rest of the computation. Note here that $(O_w^{\mathsf{str}})^4 = I$, and thus $(O_w^{\mathsf{str}})^3 = (O_w^{\mathsf{str}})^{-1}$, which is what we need to implement for our uncompute operations.

We observe that the above operation always returns x_j whenever we have a $*$-input, and y_j whenever we have a $\dagger$-input. Thus, if we run algorithm $\mathcal{B}$ with this oracle operation (in superposition, including uncomputation), then we output $f(x) = 0$ on a $*$-input, and $f(y) = 1$ on a $\dagger$-input, with the same success probability as that of $\mathcal{A}$. As this query operation can be implemented with a constant number of calls to the query oracle O_w^{str}, we conclude that $\mathsf{QS}_{\mathsf{str}}(f) = O(\mathsf{Q}(f))$. ◀

It is not obvious how we can modify the above algorithm to also output the index where x and y differ. However, we can design such an algorithm using very different techniques, and give an upper bound on $\mathsf{QS}_{\mathsf{str}}^{\mathsf{ind}}(f)$ in terms of $\mathsf{QD}(f)$. To that end, we first prove a fundamental lemma that is similar to [11, Lemma 12] and [21, Lemma 3.1].

▶ **Lemma 3.** *Let $x \in \{0,1\}^n$ be an input and let $B \subseteq [n]$. Let $\mathcal{A}$ be a T-query quantum algorithm that accepts x and rejects x_B with high probability, or more generally produces output states that are a constant distance apart in trace distance for x and x_B. Let p_t, resp. p_t^B, be the probability that, when $\mathcal{A}$ is run on x, resp. x_B, up until, but not including, the t-th query and then measured, it is found to be querying a position $i \in B$. Then,*

$$\sum_{t=1}^{T} (p_t + p_t^B) = \Omega(1).$$

Proof. We write $y = x_B$, and we let $|\psi_x^t\rangle$ and $|\psi_y^t\rangle$ be the states right before the tth query, when we run $\mathcal{A}$ on inputs x and y, respectively. We also let $|\psi_x\rangle$ and $|\psi_y\rangle$ be the final states of the algorithm run on x and y, respectively. Since $\mathcal{A}$ can distinguish x and y with high probability, we observe that $|\psi_x\rangle$ and $|\psi_y\rangle$ must be far apart, i.e., their inner product must satisfy

$$1 - |\langle \psi_x | \psi_y \rangle| = \Omega(1).$$

For every $j \in [n]$, let $\mathcal{H}_j$ be the subspace of the state space of $\mathcal{A}$ that queries the jth bit of the input. In other words, we let $\mathcal{H}_j$ be the span of all states that pick up a phase of $(-1)^{x_j}$, when the algorithm $\mathcal{A}$ calls the oracle O_x. We let Π_j be the projector on this subspace.

Next, we let $\mathcal{H}_B$ be the subspace that contains all $\mathcal{H}_j$'s with $j \in B$. In other words, we write $\mathcal{H}_B = \oplus_{j\in B}\mathcal{H}_j$. We immediately observe that the projector onto $\mathcal{H}_B$, denoted by Π_B, satisfies $\Pi_B = \sum_{j\in B}\Pi_j$. Note that it is exactly the subspace $\mathcal{H}_B$ on which the oracles O_x and O_y act differently. So, intuitively, if a state $|\psi\rangle$ has a big component in $\mathcal{H}_B$, then $O_x|\psi\rangle$ and $O_y|\psi\rangle$ will be far apart from each other.

Now, we define $p_{x,t} := \|\Pi_B|\psi_x^t\rangle\|^2$, i.e., the squared overlap of the state $|\psi_x^t\rangle$ with the subspace $\mathcal{H}_B$. Intuitively, if we were to interrupt the algorithm $\mathcal{A}$ run on input x right before the tth query, and we were to measure the query register, then the probability of measuring a $j \in B$ is $p_{x,t}$.

The crucial observation that we make is that

$$\begin{aligned}
&\left|\langle\psi_x^t|\psi_y^t\rangle\right| - \left|\langle\psi_x^{t+1}|\psi_y^{t+1}\rangle\right| \leq \left|\langle\psi_x^t|\psi_y^t\rangle - \langle\psi_x^{t+1}|\psi_y^{t+1}\rangle\right| = \left|\langle\psi_x^t|\psi_y^t\rangle - \langle\psi_x^t|O_x^\dagger O_y|\psi_y^t\rangle\right| \\
&\quad = \left|\langle\psi_x^t|(I - O_x^\dagger O_y)|\psi_y^t\rangle\right| = 2\left|\langle\psi_x^t|\Pi_B|\psi_y^t\rangle\right| \leq 2\left\|\Pi_B|\psi_x^t\rangle\right\| \cdot \left\|\Pi_B|\psi_y^t\rangle\right\| \\
&\quad = 2\sqrt{p_{x,t}\cdot p_{y,t}} \leq p_{x,t} + p_{y,t}.
\end{aligned}$$

Here, we used the triangle inequality, the Cauchy-Schwarz inequality, and the AM-GM inequality, in order. Finally, we observe that the initial states for algorithm $\mathcal{A}$ run on x and y are the same, and so $|\langle\psi_x^1|\psi_y^1\rangle| = 1$. Thus, identifying $|\psi_x^{T+1}\rangle$ with $|\psi_x\rangle$, and similarly for y, we obtain that

$$1 - |\langle\psi_x|\psi_y\rangle| = \sum_{t=1}^{T} |\langle\psi_x^t|\psi_y^t\rangle| - |\langle\psi_x^{t+1}|\psi_y^{t+1}\rangle| \leq \sum_{t=1}^{T}(p_{x,t} + p_{y,t}). \qquad \blacktriangleleft$$

We now show how the above lemma can be used to prove a connection between $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{str}}(f)$ and $\mathsf{QD}(f)$.

▶ **Theorem 17.** *Let $f : \{0,1\}^n \to \{0,1\}$. We have $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{str}}(f) = O(\mathsf{QD}(f)^{3/2})$.*

Proof. Suppose we have an algorithm $\mathcal{A}$ that distinguishes between input $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$ with high probability in T queries. Then, we construct an algorithm $\mathcal{B}$ that works in the strong sabotage input model, and finds an index $j \in [n]$ where $x_j \neq y_j$.

First, consider the following procedure. We pick an input x or y with probability $1/2$, and we pick a time step $t \in \{1,\ldots,T\}$ uniformly at random. We run $\mathcal{A}$ until right before the tth query, and then we measure the query register to obtain an index $j \in [n]$. From

Lemma 3, we obtain that the probability that $x_j \neq y_j$ is lower bounded by $\Omega(1/T)$.[2] Thus, running this algorithm $O(T)$ times would suffice to find a $j \in [n]$ such that $x_j \neq y_j$, with high probability.

However, we can do slightly better than that. Note that once the algorithm gives us an index $j \in [n]$, it takes just one query (to z) to find out if $z_j \in \{*, \dagger\}$. Thus, we can use amplitude amplification, and use $O(\sqrt{T})$ iterations of the above procedure, to find a $j \in [n]$ such that $z_j \in \{*, \dagger\}$ (equivalently, $x_j \neq y_j$). Each application of the procedure takes $O(T)$ queries to implement in the worst case. Thus, the final query complexity is $O(T^{3/2})$. ◀

Combining the above result with [11, Proposition 6], which states that $\mathsf{QD}(f) = O(\mathsf{Q}(f))$, immediately yields the following theorem.

▶ **Theorem 2.** *Let n be a positive integer, let $D \subseteq \{0,1\}^n$, and let $f : D \to \{0,1\}$ be a (partial) Boolean function. Then,* $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{str}}(f) = O(\mathsf{Q}(f)^{1.5})$.

5 $\mathsf{QS}_{\mathsf{str}}(f)$ vs. $\sqrt{\mathsf{fbs}(f)}$

So far we have shown upper bounds on quantum sabotage complexity. In this section we show our lower bounds. We first show that $\mathsf{QS}_{\mathsf{str}}(f)$ (and hence $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{str}}(f), \mathsf{QS}_{\mathsf{weak}}(f), \mathsf{QS}^{\mathsf{ind}}_{\mathsf{weak}}(f)$ as well) is bounded from below by $\sqrt{\mathsf{fbs}(f)}$. This is a generalization of the known bound of $\mathsf{Q}(f) = \Omega(\sqrt{\mathsf{bs}(f)})$ [7]. In particular, this already implies that quantum sabotage complexity is polynomially related to quantum query complexity for all total Boolean functions f. Next observe that, unlike in the usual quantum query setting, this $\sqrt{\mathsf{fbs}(f)}$ lower bound is *tight* for standard functions such as Or, And, Majority and Parity because of the Grover-based $O(\sqrt{n})$ upper bound on the quantum sabotage complexity of all functions. This suggests the possibility of the quantum sabotage complexity of f being $\Theta(\sqrt{\mathsf{fbs}(f)})$ for all total f. In the next subsection we rule this out, witnessed by f as the Indexing function, for which we show the quantum sabotage complexity to be $\Theta(\mathsf{fbs}(f))$.

5.1 A general lower bound

In the appendix we show that $\mathsf{RS}(f) = \Omega(\mathsf{fbs}(f))$, as well as the quantum bound of $\mathsf{Q}(f) = \Omega(\sqrt{\mathsf{fbs}(f)})$. We now wish to follow the same approach as in the proof of the latter bound for proving a lower bound on $\mathsf{QS}_{\mathsf{str}}(f)$. To that end, we know that $\mathsf{QS}_{\mathsf{str}}(f) := \mathsf{Q}(f^{\mathsf{str}}_{\mathsf{sab}}) = \Omega(\mathsf{ADV}^{+}(f^{\mathsf{str}}_{\mathsf{sab}}))$ [4], so it remains to show that $\mathsf{ADV}^{+}(f^{\mathsf{str}}_{\mathsf{sab}}) = \Omega(\sqrt{\mathsf{fbs}(f)})$. Thus, we adapt the proof of Lemma 24 in the sabotaged setting.

▶ **Lemma 18.** *Let n be a positive integer, $D \subseteq \{0,1\}^n$ and $f : D \to \{0,1\}$. Then,* $\mathsf{ADV}^{+}(f^{\mathsf{str}}_{\mathsf{sab}}) = \Omega(\sqrt{\mathsf{fbs}(f)})$.

Proof. Let x be the instance for which $\mathsf{fbs}(f) = \mathsf{fbs}(f,x)$, and let $(w_y)_{y \in Y}$ be the optimal weight assignment (see Definition 8). Similar to the proof of Lemma 24, we generate a (non-negative weight) adversary matrix, $\Gamma \in \mathbb{R}^{D^{\mathsf{str}}_{\mathsf{sab}} \times D^{\mathsf{str}}_{\mathsf{sab}}}$. We define Γ to be the all-zeros matrix, except for the instances where $((x,y,*),(x,y',\dagger))$ and $((x,y,\dagger),(x,y',*))$, with $y, y' \in Y$, where we define it to be

$$\Gamma[(x,y,*),(x,y',\dagger)] = \Gamma[(x,y',\dagger),(x,y,*)] = \sqrt{w_y w_{y'}}.$$

[2] It is important to remark here that we have access to our strong sabotage oracle now (Definition 14). Recall that there are four registers: $|j\rangle, |b_x\rangle, |b_y\rangle, |b_z\rangle$. When we say "run $\mathcal{A}$" with this oracle, we mean all operations act as identity on the last register.

Again Γ has a simple sparsity pattern. Only the rows and columns that are labeled by $(x, y, *)$ and $(x, y, \dagger)$, with $y \in Y$, are non-zero. Additionally, observe from the definition of the matrix entries that for any $y, y' \in Y$, we have

$$\Gamma[(x, y, *), (x, y', \dagger)] = \sqrt{w_y w_{y'}} = \Gamma[(x, y, \dagger), (x, y', *)].$$

Thus, in every 2×2-block formed by rows $(x, y, *)$ and $(x, y, \dagger)$ and columns $(x, y', *)$ and $(x, y, \dagger)$, we have that the two diagonal elements are 0, and the two off-diagonal elements are equal. Hence, by removing unimportant rows and columns that are completely zero, we can rewrite Γ as

$$\Gamma = \Gamma' \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \text{where} \qquad \Gamma' \in \mathbb{R}^{Y \times Y}, \qquad \text{with} \qquad \Gamma'[y, y'] = \sqrt{w_y w_{y'}}.$$

It now becomes apparent that Γ' is of rank 1. Indeed, it is the outer product of a vector $\sqrt{w} \in \mathbb{R}^Y$, that contains the entries $\sqrt{w_y}$ at every index labeled by y. From some matrix arithmetic, we now obtain

$$\|\Gamma\| = \|\Gamma'\| \cdot 1 = \left\|\sqrt{w}\sqrt{w}^T\right\| = \left\|\sqrt{w}\right\|^2 = \sum_{y \in Y} w_y = \mathsf{fbs}(f).$$

Thus, it remains to prove that $\|\Gamma \circ \Delta_j\| = O(\sqrt{\mathsf{fbs}(f)})$, for all $j \in [n]$. Indeed, then we can scale our matrix Γ down by $\Theta(\sqrt{\mathsf{fbs}(f)})$ so that it is feasible for the optimization program in Definition 9, and the objective value will then become $\Theta(\sqrt{\mathsf{fbs}(f)})$ as predicted.

Let $j \in [n]$. To compute $\|\Gamma \circ \Delta_j\|$, we look at its sparsity pattern. Observe that whenever we query the jth bit of $(x, y, *)$ and $(x, y', \dagger)$, we obtain the tuples (x_j, y_j, z_j) and (x_j, y'_j, z'_j). If $y_j \neq y'_j$, then it is clear that these tuples are not the same. Similarly, if $x_j \neq y_j = y'_j$, then both z_j's will be different, as one will be a $*$ and the other will be a $\dagger$. Thus, the queried tuples are only identical whenever $x_j = y_j = y'_j$, which implies that we can rewrite

$$\Gamma \circ \Delta_j = \Gamma'_j \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \text{where} \quad \Gamma'_j \in \mathbb{R}^{Y \times Y}, \quad \text{with} \quad \Gamma'_j[y, y'] = \begin{cases} 0, & \text{if } y_j = y'_j = x_j, \\ \sqrt{w_y w_{y'}}, & \text{otherwise.} \end{cases}$$

We now let $Y_0 = \{y \in Y : x_j \neq y_j\}$ and $Y_1 = \{y \in Y : x_j = y_j\}$. We interpret Γ'_j as a 2×2-block matrix, where the first rows and columns are indexed by Y_0 and the last ones are indexed by Y_1. Then, Γ'_j takes on the shape $\Gamma'_j = \begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix}$, with $A \in \mathbb{R}^{Y_0 \times Y_0}$ and $B \in \mathbb{R}^{Y_0 \times Y_1}$, defined as

$$A[y, y'] = \sqrt{w_y w_{y'}}, \qquad \text{and} \qquad B[y, y'] = \sqrt{w_y w_{y'}}.$$

We observe that

$$\|\Gamma \circ \Delta_j\| = \left\|\Gamma'_j\right\| \cdot 1 \leq \|A\| + \|B\|,$$

and hence it remains to compute $\|A\|$ and $\|B\|$.

We now define the vectors $\sqrt{w_0} \in \mathbb{R}^{Y_0}$ and $\sqrt{w_1} \in \mathbb{R}^{Y_1}$, defined by $(\sqrt{w_0})_y = \sqrt{w_y}$, and $(\sqrt{w_1})_y = \sqrt{w_y}$. We observe that $A = \sqrt{w_0}\sqrt{w_0}^T$, and $B = \sqrt{w_0}\sqrt{w_1}^T$. Thus, we obtain

$$\|A\|^2 = \left\|\sqrt{w_0}\sqrt{w_0}^T\right\|^2 = \|\sqrt{w_0}\|^4 = \left[\sum_{y \in Y_0} w_y\right]^2 = \left[\sum_{\substack{y \in Y \\ x_j \neq w_j}} w_y\right]^2 \leq 1,$$

and similarly

$$\|B\|^2 = \left\| \sqrt{w_0}\sqrt{w_1}^T \right\|^2 = \|\sqrt{w_0}\| \cdot \|\sqrt{w_1}\|^2 = \left[\sum_{\substack{y \in Y \\ x_j \neq y_j}} w_y \right] \cdot \left[\sum_{\substack{y \in Y \\ x_j = y_j}} w_y \right] \leq 1 \cdot \mathsf{fbs}(f). \qquad ◀$$

The proof of Theorem 5 now follows as a simple corollary from this lemma and the fact that $\mathsf{QS}_{\mathsf{str}}(f) := \mathsf{Q}(f^{\mathsf{str}}_{\mathsf{sab}}) = \Omega(\mathsf{ADV}^{+}(f^{\mathsf{str}}_{\mathsf{sab}}))$, where the last bound follows from the positive-weighted adversary lower bound of Ambainis [4].

5.2 A stronger lower bound for Indexing

As observed in the discussion following Theorem 2, we have $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{weak}}(f)$ (and hence $\mathsf{QS}_{\mathsf{weak}}(f)$ and $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{str}}(f)$ and $\mathsf{QS}_{\mathsf{str}}(f)$) is $O(\sqrt{n})$ for all $f : \{0,1\}^n \to \{0,1\}$. In particular, the $\sqrt{\mathsf{fbs}(f)}$ lower bound for $\mathsf{QS}_{\mathsf{str}}(f)$ is tight for standard functions like AND, OR, Parity, Majority. In view of this it is feasible that $\mathsf{QS}_{\mathsf{str}}(f) = O(\sqrt{\mathsf{fbs}(f)})$ for all total $f : \{0,1\}^n \to \{0,1\}$. In the remaining part of this section, we rule this out, witnessed by the Indexing function.

We use Ambainis' adversary method to prove lower bounds on quantum query complexity [3].

▶ **Lemma 19** ([3, Theorem 6.1]). *Let $f : \{0,1,*,\dagger\}^k \to \{0,1\}$ be a (partial) Boolean function. Let $X, Y \subseteq \{0,1,*,\dagger\}^k$ be two sets of inputs such that $f(x) \neq f(y)$ if $x \in X$ and $y \in Y$. Let $R \subseteq X \times Y$ be nonempty, and satisfy:*

- *For every $x \in X$ there exist at least m_X different $y \in Y$ such that $(x, y) \in R$.*
- *For every $y \in Y$ there exist at least m_Y different $x \in X$ such that $(x, y) \in R$.*

Let $\ell_{x,i}$ be the number of $y \in Y$ such that $(x, y) \in R$ and $x_i \neq y_i$, and similarly for $\ell_{y,i}$. Let $\ell_{\max} = \max_{i \in [k]} \max_{(x,y) \in R, x_i \neq y_i} \ell_{x,i}\ell_{y,i}$. Then any algorithm that computes f with success probability at least $2/3$ uses $\Omega\left(\sqrt{\frac{m_X m_Y}{\ell_{\max}}}\right)$ quantum queries to the input function.

Define the Indexing function as follows. For a positive integer $n > 0$, define the function $\mathsf{IND}_n : \{0,1\}^n \times \{0,1\}^{2^n} \to \{0,1\}$ as $\mathsf{IND}_n(a, b) = b_{\mathsf{bin}(a)}$, where $\mathsf{bin}(a)$ denotes the integer in $[2^n]$ whose binary representation is a.

We first note that the fractional block sensitivity is easily seen to be bounded from below by block sensitivity, and bounded from above by deterministic (in fact randomized) query complexity. Both the block sensitivity (in fact, sensitivity) and deterministic query complexity of IND_n are easily seen to be $n + 1$, implying $\mathsf{fbs}(\mathsf{IND}_n) = n + 1$.

▶ **Theorem 20.** *Let n be a positive integer. Then,* $\mathsf{QS}_{\mathsf{weak}}(\mathsf{IND}_n) = \Omega(n)$.

Proof. Recall from Definition 15 that $\mathsf{QS}_{\mathsf{weak}}(\mathsf{IND}_n) = \mathsf{Q}(\mathsf{IND}_{n,\mathsf{sab}})$. We construct a hard relation for $f = \mathsf{IND}_{n,\mathsf{sab}}$ and use Lemma 19. Recall that this relation must contain pairs of inputs. For each pair, the function must evaluate to different outputs. For ease of notation we first define the pairs of inputs $(a_1, b_1), (a_2, b_2)$ in the relation, and then justify that these inputs are indeed in S_* and $S_\dagger$, respectively.

Define $(a_1, b_1), (a_2, b_2) \in R$ if and only if all of the following hold true:

1. $(a_1, b_1) \in f^{-1}(0)$ (i.e., $(a_1, b_1) \in S_*$ for IND_n), $(x_2, y_2) \in f^{-1}(1)$ (i.e., $(a_2, b_2) \in S_\dagger$ for IND_n),
2. $|a_1 \oplus a_2| = 2$, i.e., the Hamming distance between a_1 and a_2 is 2,
3. b_1 is all-0, except for the $\mathsf{bin}(a_1)$'th index, which is $*$,
4. b_2 is all-0, except for the $\mathsf{bin}(a_2)$'th index, which is $\dagger$.

First note that $(a_1, b_1) = [(a_1, 0^{2^n}), (a_1, e_{a_1}), *]$, where $e_{a_1} \in \{0,1\}^{2^n}$ is the all-0 string except for the $\mathsf{bin}(a_1)$'th location, which is a 1. Similarly, $(a_2, b_2) = [(a_2, 0^{2^n}), (a_2, e_{a_2}), \dagger]$. Thus, (a_1, b_1) and (a_2, b_2) are in S_* and $S_\dagger$, respectively. In particular, in the language of Lemma 19 we have

$$X = \left\{(a,b) \in \{0,1\}^n \times \{0,1,*\}^{2^n} : b \text{ is all-0 except for the } \mathsf{bin}(a)\text{'th index, which is } *\right\}. \tag{5}$$

Similarly,

$$Y = \left\{(a,b) \in \{0,1\}^n \times \{0,1,\dagger\}^{2^n} : b \text{ is all-0 except for the } \mathsf{bin}(a)\text{'th index, which is } \dagger\right\}. \tag{6}$$

We now analyze the quantities m_X, m_Y and $\ell_{\max}$ from Lemma 19.

1. $m_X = m_Y = \binom{n}{2}$: Consider $(a,b) \in X$. The number of elements $(a',b') \in Y$ such that $((a,b),(a',b') \in R)$ is simply the number of strings a' that have a Hamming distance of 2 from a, since each such a' corresponds to exactly one b' with $((a,b),(a',b') \in R)$, where b' is the all-0 string except for the $\mathsf{bin}(a)'$th location which is a $\dagger$. Thus $m_X = \binom{n}{2}$. The argument for m_Y is essentially the same.
2. $\ell_{\max} = \min\left\{\binom{n}{2}, (n-1)^2\right\}$: We consider two cases.
   - **a.** $i \in [n]$: Fix $((a,b),(a',b')) \in R$ with $a_i \neq a'_i$. Recall that $\ell_{(a,b),i}$ is the number of $(a'',b'') \in Y$ such that $((a,b),(a'',b'')) \in R$ and $a_i \neq a''_i$. Following a similar logic as in the previous argument, this is simply the number of a'' that have Hamming distance 2 from a and additionally satisfy $a_i \neq a''_i$. There are $n-1$ possible locations for the other difference between a and a'', so $\ell_{(a,b),i} = n-1$ in this case. Essentially the same argument shows $\ell_{(a',b'),i} = n-1$, and so $\ell_{(a,b),i} \cdot \ell_{(a',b'),i} = (n-1)^2$.
   - **b.** $i \in [2^n]$: Fix $((a,b),(a',b')) \in R$ with $b_i \neq b'_i$. By the structure of R, b is the all-0 string except for the $\mathsf{bin}(a)$'th location which is a $*$, and b' is the all-0 string except for the $\mathsf{bin}(a')$'th location which is a $\dagger$. Thus $i \in \{\mathsf{bin}(a), \mathsf{bin}(a')\}$. Without loss of generality, assume $i = \mathsf{bin}(a)$, and thus $b_{\mathsf{bin}(a)} = *$. For each a'' with Hamming distance 2 from a, we have $((a,b),(a'',b'')) \in R$ where b'' is all-0 except for the a''th index which is $\dagger$. In particular, $b''_{\mathsf{bin}(a)} = 0$. So we have $\ell_{(a,b),i} = \binom{n}{2}$. On the other hand, we have $b'_{\mathsf{bin}(a)} = 0$. So the only (a'',b'') with $((a'',b''),(a',b')) \in R$ and with $b''_{\mathsf{bin}(a)} \neq b'_{\mathsf{bin}(a)}$ is $(a'',b'') = (a,b)$. Thus $\ell_{(a',b'),i} = 1$.

Lemma 19 then implies

$$\mathsf{QS}_{\mathsf{weak}}(\mathsf{IND}_n) = \mathsf{Q}(\mathsf{IND}_{n,\mathsf{sab}}) = \mathsf{Q}(f) = \Omega\left(\sqrt{\frac{\binom{n}{2}^2}{\min\left\{\binom{n}{2}, (n-1)^2\right\}}}\right) = \Omega(n), \tag{7}$$

proving the theorem. ◀

This proof can easily be adapted to also yield a lower bound of $\mathsf{QS}_{\mathsf{str}}(\mathsf{IND}_n) = \Omega(n)$. We include a proof in the appendix for completeness. As a corollary we obtain the following.

▶ **Corollary 21.** *Let n be a positive integer. Then, $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{str}}(\mathsf{IND}_n) = \Omega(n)$ and $\mathsf{QS}^{\mathsf{ind}}_{\mathsf{weak}}(\mathsf{IND}_n) = \Omega(n)$.*

6 Open questions

In this paper we studied the quantum sabotage complexity of Boolean functions, which we believe is a natural extension of randomized sabotage complexity introduced by Ben-David and Kothari [10]. We note here that in a subsequent work [11] they also defined a quantum analog, but this is fairly different from the notion that we studied in this paper.

We argued, by showing several results, that it makes sense to consider four different models depending on the access to input, and output requirements. While it is easily seen that the randomized sabotage complexity of a function remains (asymptotically) the same regardless of the choice of model (see Proposition 22), such a statement is not clear in the quantum setting. In our view, the most interesting problem left open from our work is to prove or disprove that even the quantum sabotage complexity of a function is asymptotically the same in all of these four models. It would also be interesting to see tight polynomial relationships between the various quantum sabotage complexities and quantum query complexity.

References

1 Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 863–876. ACM, 2016. doi:10.1145/2897518.2897644.

2 Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of huang's sensitivity theorem. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1330–1342. ACM, 2021. doi:10.1145/3406325.3451047.

3 Andris Ambainis. Quantum lower bounds by quantum arguments. *J. Comput. Syst. Sci.*, 64(4):750–767, 2002. Earlier version in STOC'00. doi:10.1006/JCSS.2002.1826.

4 Andris Ambainis. Polynomial degree vs. quantum query complexity. *J. Comput. Syst. Sci.*, 72(2):220–238, 2006. doi:10.1016/J.JCSS.2005.06.006.

5 Anurag Anshu, Dmitry Gavinsky, Rahul Jain, Srijita Kundu, Troy Lee, Priyanka Mukhopadhyay, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity. In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 93 of *LIPIcs*, pages 10:1–10:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.FSTTCS.2017.10.

6 Andrew Bassilakis, Andrew Drucker, Mika Göös, Lunjia Hu, Weiyun Ma, and Li-Yang Tan. The power of many samples in query complexity. In *47th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 168 of *LIPIcs*, pages 9:1–9:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ICALP.2020.9.

7 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. Earlier version in FOCS'98. doi:10.1145/502090.502097.

8 Shalev Ben-David and Eric Blais. A tight composition theorem for the randomized query complexity of partial functions: Extended abstract. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 240–246. IEEE, 2020. doi:10.1109/FOCS46700.2020.00031.

9 Shalev Ben-David, Eric Blais, Mika Göös, and Gilbert Maystre. Randomised composition and small-bias minimax. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 624–635. IEEE, 2022. doi:10.1109/FOCS54457.2022.00065.

10 Shalev Ben-David and Robin Kothari. Randomized query complexity of sabotaged and composed functions. *Theory Comput.*, 14(1):1–27, 2018. Earlier version in ICALP'16. doi:10.4086/TOC.2018.V014A005.

11 Shalev Ben-David and Robin Kothari. Quantum distinguishing complexity, zero-error algorithms, and statistical zero knowledge. In *14th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC*, volume 135 of *LIPIcs*, pages 2:1–2:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.TQC.2019.2.

12 Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997. doi:10.1137/S0097539796300933.

13 Sourav Chakraborty, Chandrima Kayal, Rajat Mittal, Manaswi Paraashar, Swagato Sanyal, and Nitin Saurabh. On the composition of randomized query complexity and approximate degree. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, volume 275 of *LIPIcs*, pages 63:1–63:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.APPROX/RANDOM.2023.63.

14 Dmitry Gavinsky, Troy Lee, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity via max-conflict complexity. In *46th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 132 of *LIPIcs*, pages 64:1–64:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.ICALP.2019.64.

15 Justin Gilmer, Michael E. Saks, and Srikanth Srinivasan. Composition limits and separating examples for some boolean function complexity measures. *Comb.*, 36(3):265–311, 2016. doi:10.1007/S00493-014-3189-X.

16 Mika Göös, T. S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication versus partition number. *ACM Trans. Comput. Theory*, 10(1):4:1–4:20, 2018. Earlier version in ICALP'17. doi:10.1145/3170711.

17 Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 212–219. ACM, 1996. doi:10.1145/237814.237866.

18 Peter Høyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 526–535. ACM, 2007. doi:10.1145/1250790.1250867.

19 Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019.

20 Raghav Kulkarni and Avishay Tal. On fractional block sensitivity. *Chic. J. Theor. Comput. Sci.*, 2016, 2016. URL: http://cjtcs.cs.uchicago.edu/articles/2016/8/contents.html.

21 Sophie Laplante and Frédéric Magniez. Lower bounds for randomized and quantum query complexity using kolmogorov arguments. *SIAM J. Comput.*, 38(1):46–62, 2008. doi:10.1137/050639090.

22 Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion. In *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 344–353. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.75.

23 Noam Nisan. CREW prams and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991. Earlier version in STOC'89. doi:10.1137/0220062.

24 Swagato Sanyal. Randomized query composition and product distributions. In *41st International Symposium on Theoretical Aspects of Computer Science, STACS*, volume 289 of *LIPIcs*, pages 56:1–56:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.STACS.2024.56.

25 Robert Spalek and Mario Szegedy. All quantum adversary methods are equivalent. *Theory Comput.*, 2(1):1–18, 2006. doi:10.4086/TOC.2006.V002A001.

26 Ronald de Wolf. Quantum computing: Lecture notes, 2023. Version 5. arXiv:1907.09415.

A Randomized sabotage complexity

By comparing the definitions in Definition 15 to those in [10], we observe that $\mathsf{RS} = \mathsf{RS}_{\mathsf{weak}}$. However, we argue that for all functions, the other randomized complexity measures are equal up to constants.

▶ **Proposition 22.** *Let n be a positive integer, and $D \subseteq \{0,1\}^n$. Let $f : D \to \{0,1\}$ be a (partial) Boolean function. Then,*

$$\mathsf{RS}(f) := \mathsf{RS}_{\mathsf{weak}}(f) = \Theta(\mathsf{RS}_{\mathsf{str}}(f)) = \Theta(\mathsf{RS}^{\mathsf{ind}}_{\mathsf{weak}}(f)) = \Theta(\mathsf{RS}^{\mathsf{ind}}_{\mathsf{str}}(f)).$$

Proof. It is clear that we have $\mathsf{RS}_{\mathsf{str}}(f) \leq \mathsf{RS}_{\mathsf{weak}}(f)$, and $\mathsf{RS}^{\mathsf{ind}}_{\mathsf{str}}(f) \leq \mathsf{RS}^{\mathsf{ind}}_{\mathsf{weak}}(f)$ because the input model is strictly stronger. For the opposite directions, suppose that we have an algorithm $\mathcal{A}$ in the strong input model. Let $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$. Note that for every $j \in [n]$ on queries where $x_j = y_j$, the oracle outputs $(0,0,0)$ or $(1,1,1)$. Thus, on these indices, the same algorithm in the weak model would yield the outputs 0 or 1, respectively. Moreover, with high probability $\mathcal{A}$ must query a $j \in [n]$ where $x_j \neq y_j$ at some point, since otherwise it cannot distinguish $[x,y,*]$ and $[x,y,\dagger]$. This gives us the following $\mathsf{RS}^{\mathsf{ind}}_{\mathsf{weak}}$ algorithm: run $\mathcal{A}$ constantly many times (with strong queries replaced by weak queries) until we query a $j \in [n]$ where $x_j \neq y_j$. At that point, we can interrupt the algorithm and we have enough information to solve the sabotage problem, in both the decision and index model. Thus $\mathsf{RS}_{\mathsf{weak}}(f) = O(\mathsf{RS}_{\mathsf{str}}(f))$, and $\mathsf{RS}^{\mathsf{ind}}_{\mathsf{weak}}(f) = O(\mathsf{RS}^{\mathsf{ind}}_{\mathsf{str}}(f))$.

It remains to show that $\mathsf{RS}_{\mathsf{str}}(f) = \Theta(\mathsf{RS}^{\mathsf{ind}}_{\mathsf{str}}(f))$. By the definitions of the models, we have $\mathsf{RS}_{\mathsf{str}}(f) \leq \mathsf{RS}^{\mathsf{ind}}_{\mathsf{str}}(f)$. On the other hand, suppose that we have an algorithm $\mathcal{B}$ that figures out whether we have a $*$- or $\dagger$-input. Then, by the same logic as given in the previous paragraph, with high probability, $\mathcal{B}$ must have encountered at least one $*$ or $\dagger$, so we can read back in the transcript to find out at which position it made that query. Repeating $\mathcal{B}$ a constant number of times this way yields $\mathsf{RS}^{\mathsf{ind}}_{\mathsf{str}}(f) = O(\mathsf{RS}_{\mathsf{str}}(f))$. ◀

Note that we did not do a formal analysis of success probabilities in the above argument, but this is not hard to do. We omit precise details for the sake of brevity.

B Lower bounds on $\mathsf{QS}(f)$ and $\mathsf{RS}(f)$ in terms of $\mathsf{fbs}(f)$

To the best of our knowledge, the only lower bound on randomized sabotage complexity $\mathsf{RS}(f)$ in the existing literature is $\Omega(\mathsf{QD}(f))$ [11, Corollary 11]. We remark that $\mathsf{RS}(f)$ can also be lower bounded by $\mathsf{fbs}(f)$. This result was essentially shown in [10, Theorem 7.2], but we include a proof here for completeness.

▶ **Lemma 23.** $\mathsf{fbs}(f) = O(\mathsf{RS}(f))$.

Proof. From [10, Theorem 3.3], we find that $\mathsf{R}(f_{\mathsf{sab}}) = \Theta(\mathsf{RS}(f))$, and so it suffices to show that $\mathsf{fbs}(f) \leq 10\mathsf{R}(f_{\mathsf{sab}})$. By Yao's minimax principle, it suffices to exhibit a distribution over inputs for which any deterministic algorithm fails to compute f_{sab} correctly with probability at least $2/3$ in less than $\mathsf{fbs}(f)/10$ queries.

Without loss of generality, let $x \in f^{-1}(0)$ be the instance for which fractional block-sensitivity is maximized, let $Y = f^{-1}(1)$, and let $(w_y)_{y \in Y}$ be the optimal weight assignment in the fractional block sensitivity linear program (see Definition 8). We can similarly think of every $y \in Y$ as obtained by flipping the bits in x that belong to a sensitive block B. We denote $y = x_B$, and write $w_y = w_B$.

Let T be a deterministic algorithm for f_{sab} with query complexity less than $\mathsf{fbs}(f)/10$, and let μ be a distribution defined as follows: for all $B \subseteq [n]$ that is a sensitive block for f, let $\mu([x, x_B, *]) = \mu([x, x_B, \dagger]) = w_B/(2\mathsf{fbs}(f))$. Since $\sum_{B\subseteq[n]} w_B = \mathsf{fbs}(f)$ where the sum is over all sensitive blocks for f, μ forms a legitimate probability distribution.

Consider a leaf L of T, and suppose that the output at L is $b \in \{0, 1\}$. By the definition of fractional block sensitivity, we know that for any $j \in [n]$, we have $\sum_{B:j\in B} w_B \leq 1$. Thus, by a union bound, the μ-mass of inputs supported by our distribution (of the form $[x, x_B, *]$ or $[x, x_B, \dagger]$) reaching L, and such that no bit of B has been read on this path is at least $1 - \left(\frac{\mathsf{fbs}(f)}{10} \cdot \frac{1}{\mathsf{fbs}(f)}\right) \geq 9/10$. Note that for each such B, both the input $[x, x_B, *]$ and $[x, x_B, \dagger]$ reach L. Since μ allotted equal mass to each such pair, this means an error is made at L for inputs with μ-mass at least $9/20 > 1/3$, concluding the proof. ◀

We now turn to lower bounding $\mathsf{QS}_{\mathsf{str}}(f)$ by $\Omega(\sqrt{\mathsf{fbs}(f)})$. To that end, recall that for any Boolean function f, we have the chain of inequalities $\mathsf{Q}(f) = \Omega(\mathsf{ADV}^+(f)) = \Omega(\sqrt{\mathsf{fbs}(f)})$. The first inequality can be found in [25], for example. While we believe the second inequality is known, we were again unable to find a published proof of it. We include a proof here for completeness.

▶ **Lemma 24.** *Let n be a positive integer, $D \subseteq \{0, 1\}^n$, and $f : D \to \{0, 1\}$ be a (partial) Boolean function. Then, $\sqrt{\mathsf{fbs}(f)} = O(\mathsf{ADV}^+(f))$.*

Proof. Recall the definition of fractional block sensitivity, Definition 8. Let x be the input to f for which $\mathsf{fbs}(f, x) = \mathsf{fbs}(f)$. Let $(w_y)_{y\in Y}$ be the optimal solution of the linear program. We now define an adversary matrix Γ, that forms a feasible solution to the optimization program in Definition 9. In particular, we define $\Gamma \in \mathbb{R}^{D\times D}$ as the all-zeros matrix, except for the entries (x, y) and (y, x) where $y \in Y$, which we define to be

$$\Gamma[x, y] = \Gamma[y, x] = \sqrt{w_y}.$$

The matrix Γ we constructed is very sparse. It is only non-zero in the row and column that is indexed by x. Moreover, $\Gamma[x, x] = 0$ as well. In particular, this makes computing its norm quite easy, we just have to compute the ℓ_2-norm of the column indexed by x.

For any $j \in [n]$, we thus find

$$\|\Gamma \circ \Delta_j\|^2 = \sum_{\substack{y\in Y \\ x_j\neq y_j}} \Gamma[x, y]^2 = \sum_{\substack{y\in Y \\ x_j\neq y_j}} w_y \leq 1,$$

and so Γ is a feasible solution to the optimization program in Definition 9. Finally, we have

$$\mathsf{ADV}^+(f)^2 \geq \|\Gamma\|^2 = \sum_{y\in Y} \Gamma[x, y]^2 = \sum_{y\in Y} w_y = \mathsf{fbs}(f).$$ ◀

C Missing proof

In Theorem 20 we showed using the adversary method that $\mathsf{QS}_{\mathsf{weak}}(\mathsf{IND}_n) = \Omega(n)$. We now show that the same proof can be adapted to show the same lower bound on $\mathsf{QS}_{\mathsf{str}}(\mathsf{IND}_n)$ as well.

▶ **Corollary 25.** *Let n be a positive integer. Then, $\mathsf{QS}_{\mathsf{str}}(\mathsf{IND}_n) = \Omega(n)$.*

Proof. Let $N = n + 2^n$. In the proof of Theorem 20 we constructed a hard relation R for $f = \mathsf{IND}_{n,\mathsf{sab}}$ to show a $\Omega(n)$ lower bound in the weak sabotage model. Using R we construct a hard relation for $\mathsf{IND}^{\mathsf{str}}_{n,\mathsf{sab}}$, which we denote by R^{str} in the strong sabotage model, and use Lemma 19. Define $((x, y, *), (x', y', \dagger)) \in R^{\mathsf{str}}$ if and only if all of the following hold true:[3]

1. $(x, y, *) := ((x_j, y_j, z_j))_{j=1}^N \in S^{\mathsf{str}}_*$ for IND_n, $(x', y', \dagger) := ((x'_j, y'_j, z'_j))_{j=1}^N \in S^{\mathsf{str}}_\dagger$ for IND_n,
2. $(z, z') \in R$.

Following the language of Lemma 19 we have

$$X^{\mathsf{str}} = \{(x, y, *) := ((x_j, y_j, z_j))_{j=1}^N : z \in X \text{ as defined in Equation 5}\}. \tag{8}$$

Similarly,

$$Y^{\mathsf{str}} = \{(x, y, \dagger) := ((x_j, y_j, z_j))_{j=1}^N : z \in Y \text{ as defined in Equation 6}\}. \tag{9}$$

We now analyze the quantities $m_{X^{\mathsf{str}}}$, $m_{Y^{\mathsf{str}}}$ and $\ell_{\max}$ from Lemma 19.

1. As described above, the way we construct R^{str} using elements of R ensures that $m_{X^{\mathsf{str}}} \geq m_X$ and $m_{Y^{\mathsf{str}}} \geq m_Y$. Additionally, for every $z \in X \cup Y$ there is exactly one pair in $\mathsf{IND}^{-1}(0) \times \mathsf{IND}^{-1}(1)$ for which z is the sabotaged input. Hence, $m_{X^{\mathsf{str}}} = m_X = \binom{n}{2}$ and $m_{Y^{\mathsf{str}}} = m_Y = \binom{n}{2}$.
2. $\ell_{\max} = \max\left\{\binom{n}{2}, (n-1)^2\right\}$: we consider two cases.
   a. $i \in [n]$: Fix an $(x, y, *), (x', y', \dagger) \in R^{\mathsf{str}}$ differing at an index $i \in [n]$. Recall that $\ell_{(x,y,*),i}$ denotes the number of $(x', y', \dagger) \in Y^{\mathsf{str}}$ such that $(x, y, *), (x', y', \dagger) \in R^{\mathsf{str}}$ and $(x_i, y_i, z_i) \neq (x'_i, y'_i, z'_i)$. The construction of sets $X^{\mathsf{str}}, Y^{\mathsf{str}}$ (which is in turn based on X, Y, respectively) ensures that for all $i \in [n]$ we have $x_i = y_i = z_i$ and $x'_i = y'_i = z'_i$. Hence, directly using the same arguments as in Item 2a, we get $\ell_{(x,y,*),i} = n - 1$ and $\ell_{(x',y',\dagger),i} = n - 1$, hence $\ell_{(x,y,*),i} \cdot \ell_{(x',y',\dagger),i} = (n-1)^2$.
   b. $i \in [n+1, N]$: Fix an $(x, y, *) := ((x_j, y_j, z_j))_{j=1}^N, (x', y', \dagger) := ((x'_j, y'_j, z'_j))_{j=1}^N \in R^{\mathsf{str}}$ differing at an index $i \in [n+1, N]$. By the structure of R and by extension of R^{str}, for strings $z := (a, b)$ and $z' := (a', b')$ the string b is all-0 string except for the $\mathsf{bin}(a)$'th location which is a $*$, and string b' is all-0 string except for the $\mathsf{bin}(a')$'th location which is a $\dagger$. Thus the only important case is for $i \in \{n + \mathsf{bin}(a), n + \mathsf{bin}(a')\}$. Without loss of generality, assume $i = n + \mathsf{bin}(a)$, and thus $z_i = b_{\mathsf{bin}(a)} = *$. Moreover, for every $z \in X \cup Y$ there is exactly one pair in $\mathsf{IND}^{-1}(0) \times \mathsf{IND}^{-1}(1)$ that sabotage as z. So, we have $\ell_{z,i} = \binom{n}{2}$ and $\ell_{z',i} = 1$ as $z'_i = b'_{\mathsf{bin}(a')} = 0$. Therefore, $\ell_{z,i} \cdot \ell_{z',i} = \binom{n}{2}$.

Lemma 19 then implies

$$\mathsf{QS}_{\mathsf{str}}(\mathsf{IND}_n) = \mathsf{Q}(\mathsf{IND}^{\mathsf{str}}_{n,\mathsf{sab}}) = \Omega\left(\sqrt{\frac{\binom{n}{2}^2}{\max\left\{\binom{n}{2}, (n-1)^2\right\}}}\right) = \Omega(n). \tag{10}$$

◀

[3] Recall that $(x, y, *)$ denotes $((x_j, y_j, z_j))_{j=1}^N$ where $z = [x, y, *]$ and $(x, y, \dagger)$ denotes $((x_j, y_j, z_j))_{j=1}^N$ where $z = [x, y, \dagger]$; see Definition 12.