Electronic Colloquium on Computational Complexity, Report No. 2 (2012)

Query Complexity and Error Tolerance of Witness Finding Algorithms

Akinori Kawachi, Benjamin Rossman, and Osamu Watanabe
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
E-mail: watanabe(at)is.titech.ac.jp

Abstract

We propose an abstract framework for studying search-to-decision reductions for NP. Specifically, we study the following *witness finding problem*: for a hidden nonempty set $W \subseteq \{0,1\}^n$, the goal is to output a witness in W with constant probability by making randomized queries of the form "is $Q \cap W$ nonempty?" where $Q \subseteq \{0,1\}^n$. Algorithms for the witness finding problem can be seen as a general form of search-to-decision reductions for NP. This framework is general enough to express the average-case search-to-decision reduction of Ben-David et al., as well as the Goldreich-Levin algorithm from cryptography.

Our results show that the witness finding problem requires $\Omega(n^2)$ queries with the error-free oracle, matching the upper bound of Ben-David et al. We also give a new $O(n^2)$ query algorithm that achieves an improved error tolerance of $O(1/n)$. Further results show that, in the unique-witness setting where $|W| = 1$, the query complexity and the list size of Goldreich-Levin algorithm can be improved to $O(n/\varepsilon^2)$ and $O(1/\varepsilon)$ respectively with $(1/2-\varepsilon)$-error bounded oracle, and that this is optimal up to a constant factor: For any $\varepsilon = n^{-\Omega(1)}$, with $(1/2-\varepsilon)$-error bounded oracle it is impossible to improve the query complexity to $c_1 n/\varepsilon^2$ query complexity for some small c_1 while keeping the list size polynomially bounded.

1 Introduction

We propose an abstract framework for studying the relationship between the search and decision versions of NP problems. As a generalization of search-to-decision reductions, we study the following *witness finding problem*: for a hidden nonempty subset W of $\{0,1\}^n$, the goal is to produce an element in W by asking NP-type queries of the form "is $Q \cap W$ nonempty?" where $Q \subseteq \{0,1\}^n$. Algorithms solving the witness finding problem can be seen as generic search-to-decision reductions that apply to any NP problem. This witness finding problem is also relevant to cryptography. The algorithms of Ben-David, Chor, Goldreich, and Luby (hereafter, Ben-David et al.) [BCGL92] and Goldreich and Levin [GL89] can be seen as witness finding algorithms. We discuss the query complexity of these algorithms. We also consider the situation where queries may be answered incorrectly, and we study witness finding algorithms from the perspective of error tolerance as well.

The complexity class NP is characterized as the class of sets with a polynomial-size and polynomial-time checkable witness system. That is, for any set $L \subseteq \{0,1\}^*$, it is in NP if and only if it is characterized by

$$L \;=\; \{\; x \mid \exists w \text{ such that } |w| \le q(|x|) \text{ and } R(x,w) \;\}$$

with some polynomial q and some polynomial-time computable predicate R. For any set $L \in \mathrm{NP}$, a pair (q,R) characterizing L as above is called a *witness system*, and for any $x \in L$, a string w satisfying $|w| \le q(|x|) \wedge R(x,w)$ is called a *witness* (for $x \in L$). The *decision problem* for L

ISSN 1433-8092

is the task of deciding, for a given $x \in \{0,1\}^n$, whether there exists some witness w for $x \in L$. The *search problem* for L is the task of producing a witness for a given $x \in L$.

The relationship between decision and search problems has been investigated in several contexts. In order to be specific, let us consider the 3SAT problem and the standard witness system that uses, for any 3CNF formula φ, a satisfying assignment as a witness for $\varphi \in$ 3SAT. Here the search problem is the task of finding a satisfying assignment for a given $\varphi \in$ 3SAT. A question that has been often asked is how to compute *one of* the satisfying assignments of φ by asking queries to 3SAT. It is easy to obtain a satisfying assignment for $\varphi \in$ 3SAT (e.g. the lexicographically first one) deterministically by asking queries to 3SAT *adaptively.* That is, the search problem for 3SAT is P^{NP}-computable. On the other hand, Ben-David et al. [BCGL92] used the "isolation technique" to give a randomized algorithm that solves this NP-type search problem in polynomial-time by asking queries to 3SAT *nonadaptively*; that is, the witness finding problem for 3SAT is $\mathrm{ZPP}_{||}^{\mathrm{NP}}$-computable. In this paper, we focus on the query complexity of such computations. Consider any $\varphi \in$ 3SAT with n variables. Then a witness, i.e. satisfying assignment, can be expressed as a binary string of length n. For computing such a witness, the above mentioned P^{NP}-algorithm can be implemented with n adaptive queries, whereas $O(n^2)$ nonadaptive queries are needed in the $\mathrm{ZPP}_{||}^{\mathrm{NP}}$-algorithm. We investigate in this paper whether this difference in query complexity is inherent between the adaptive and nonadaptive ways of asking queries.

As an abstract framework for discussing this type of query complexity, we introduce the witness finding problem, a problem of searching for a witness in an unknown set $W \subseteq \{0,1\}^n$ (as a generic NP-type search problem) by using queries of the form "is $Q \cap W$ nonempty?" for some $Q \subseteq \{0,1\}^n$ (as instances of a generic NP-type decision problem). Here the set W is called a *witness set* and every $w \in W$ is regarded as a *witness*; on the other hand, the set Q is regarded as a specification of a *query.* A *randomized algorithm* for solving the witness finding problem in this abstract setting is a pair $(\mathcal{Q}, \mathcal{F})$, where $\mathcal{Q}$ is a procedure to generate queries $Q_1, Q_2, ..., Q_m$ (either adaptively or nonadaptively) and $\mathcal{F} : \{0,1\}^m \to \{0,1\}^n$ is a procedure to give a witness in W based on answers to these queries. Let $A_W : 2^{\{0,1\}^n} \to \{0,1\}$ denote the oracle answering decision queries. An algorithm is successful if it produces a witness in W with constant probability for every witness set W. That is, W (or more precisely, the oracle A_W) is considered as a blackbox. By contrast, an algorithm for solving a concrete NP-type search problem only needs to succeed on a specific class of witness sets. However, even in our abstract setting, both the standard P^{NP}-algorithm and the $\mathrm{ZPP}_{||}^{\mathrm{NP}}$-algorithm of Ben-David et al. succeed in solving the witness finding problem every witness set W. In particular, the P^{NP}-algorithm uses only n queries whereas the $\mathrm{ZPP}_{||}^{\mathrm{NP}}$-algorithm uses $O(n^2)$ queries. On the other hand, we prove that any randomized nonadaptive query algorithm that solves the witness finding problem for every $W \subseteq \{0,1\}^n$ (with probability $\Omega(1)$) needs to ask $\Omega(n^2)$ queries. That is, in this abstract framework, we show that the nonadaptive query complexity for the witness finding problem is $\Theta(n^2)$ and illustrate the difference between adaptive and nonadaptive query complexity.

Valiant and Vazirani [VV86] gave a procedure, known as the "isolation technique", to modify a given NP instance to some other NP instance that reduces the number of witnesses to exactly one with a certain probability. To be specific, consider 3SAT. The isolation technique of Valiant-Vazirani is a randomized reduction f_{iso} from 3SAT to L_{iso} with the following property:

each instance $\phi \in$ 3SAT (that may have more than one satisfying assignment) is reduced to $\psi = f_{\text{iso}}(\phi)$ such that, with a certain probability, $\psi \in L_{\text{iso}}$ and this is witnessed by exactly one satisfying assignment of ψ. Recently, Kabanets and Watanabe [KW11] showed the optimality of the isolation technique of Valiant-Vazirani. They showed that the "isolation probability", i.e. the probability of a unique satisfying assignment, is at most $O(1/n)$ by any randomized reduction under a certain blackbox computation model. While the isolation technique of Valiant-Vazirani is a many-one reduction, the witness finding algorithm of Ben-David et al. [BCGL92] can be regarded as a more general truth-table reduction, and by this generalized reduction, we can achieve $1 - o(1)$ isolation probability. On the other hand, in Section 3 we show a limitation of this type of isolation by proving under a similar blackbox model that any truth-table reduction type isolation needs $\Omega(n^2)$ queries.

We also consider the situation where queries may be answered incorrectly. In fact, the motivation of Ben-David et al. was to solve a given witness finding problem *on average* by using a polynomial-time algorithm $\mathcal{A}$ that solves the corresponding NP-type decision problem with high probability on random instances. The algorithm $\mathcal{A}$ can be seen as an erroneous oracle; on the other hand, the algorithm of Ben-David et al. is tolerant against a small fraction of errors and solves the witness finding problem with high probability by using $\mathcal{A}$ as an oracle. We investigate the question of whether the algorithm of Ben-David et al. is optimal also in terms of its error tolerance. For this, we extend our abstract framework to include a notion of "ε-error tolerance." The algorithm of Ben-David et al. is $O(1/n^2)$-error tolerant in our framework. In Section 3 we show how to improve this error tolerance parameter by presenting a randomized nonadaptive algorithm that is $O(1/n)$-error tolerant while still making only $O(n^2)$ queries.

Finally, in Section 4 we consider the setting where the witness is unique, that is, the witness set consists of one element. This situation is typical for solving decoding problems. For example, for analyzing the hardcore property of one-way functions, Goldreich and Levin gave a randomized polynomial-time algorithm that solves the unique witness finding problem by using some NP oracle; see, e.g., [AB09, Gol01]. For any singleton witness set $W = \{w\}$, the Goldreich-Levin algorithm gives a list of candidates for this witness w by making queries to an erroneous oracle whose error probability is bounded by $1/2 - \varepsilon$. This may be regarded as a "list decoding algorithm" where the unique witness w is a "message" and the erroneous oracle is a "corrupted codeword". We again would like to discuss the limitation of such algorithms. For this, we relax our notion of an abstract witness finding algorithm to allow the output to be a list of candidate witnesses. It is easy to see that, for any $\varepsilon = n^{-O(1)}$, the Goldreich-Levin algorithm achieves $(1/2 - \varepsilon)$-error tolerance with $O(n^2/\varepsilon^2)$ query complexity and $O(n/\varepsilon^2)$ list size. We first show that these complexity parameters can be improved (though its polynomial-time computability is not guaranteed) to $O(n/\varepsilon^2)$ query complexity and $O(1/\varepsilon^2)$ list size. Then we show that this query complexity is close to the optimal in the following sense: There exists some small $c_1 > 0$ such that for any $\varepsilon = n^{-O(1)}$, no randomized query algorithm (even an adaptive one) exists with $c_1 n/\varepsilon^2$ query complexity that has $(1/2 - \varepsilon)$-error tolerance and polynomially bounded list size. This can be also interpreted as a lower bound of the query complexity for locally list decoding algorithms.

2 Preliminaries

We use standard notions and notations in computational complexity theory; see e.g. [AB09] for definitions. Throughout this paper, we assume strings are encoded in $\{0,1\}^*$, and we use $\Omega_n = \{0,1\}^n$ (or more simply Ω when n is clear by the context) to denote the universe of witnesses for a given length parameter n. For any set X and any distribution $\mathcal{D}$ on X, by "$\Pr_{x:\mathcal{D}}[\Phi(x)]$" we mean the probability that $\Phi(x)$ holds when x is chosen under the distribution $\mathcal{D}$. When $\mathcal{D}$ is the uniform distribution on X, we simply write it as "$\Pr_{x:X}[\Phi(x)]$."

Definition 1. A *witness set* is a nonempty subset of Ω_n denoted by W, and each element of W is a *witness.* A *query* is a set $Q \subseteq \Omega_n$ interpreted as the question "is $Q \cap W$ nonempty?" Let $A_W(Q) \in \{0,1\}$ denote the answer from the (error-free) oracle to the query Q; that is, $A_W(Q) = 1$ if and only if $Q \cap W \neq \emptyset$.

The *witness finding problem* is the problem of obtaining any one of the witnesses in W by asking queries and using oracle answers to those queries. Here we define the following abstract notion of "algorithm" for this task.

Definition 2. A *randomized witness finding query algorithm* (hereafter, we omit "randomized" and "query" for simplicity) is a pair $(\mathcal{Q}, \mathcal{F})$ of randomized algorithms where, for every witness length n and random seed $s \in \{0,1\}^{r(n)}$, $\mathcal{Q}$ produces a sequence of queries $Q_1, \ldots, Q_{m(n)} \subseteq \Omega_n$ (where Q_i may depend on the oracle answers to queries $Q_1, \ldots, Q_{i-1}$) and $\mathcal{F}$ outputs an element in Ω_n based on the answers to queries $Q_1, \ldots, Q_{m(n)}$. For a witness set W, the algorithm *succeeds* with respect to W whenever $\mathcal{F}$ correctly outputs an element of W. The *success probability* of the algorithm is the probability $\Pr_{s:\{0,1\}^{r(n)}}[\,\mathcal{F}(s, \widetilde{A}_W(\mathcal{Q}))$ succeeds$\,]$ for the worst witness set. Parameters $r(n)$ and $m(n)$ are called the *seed length* and *query complexity.*

We may sometimes use the term "success probability" in more general way meaning the probability $\Pr[\,\mathcal{F}(s, \widetilde{A}_W(\mathcal{Q}))$ succeeds$\,]$ under some distribution defined in each context. An algorithm is said to be *nonadaptive* if the distribution of queries $Q_1, \ldots, Q_m$ does not depend on the answers to these queries; otherwise, the algorithm is *adaptive.*

When n is fixed, we write simply r and m. We write $A_W(\mathcal{Q})$ as a shorthand for the sequence $A_W(Q_1), \ldots, A_W(Q_m)$ of answers to queries $Q_1, \ldots, Q_m$ issued by $\mathcal{Q}$, and we write $\mathcal{F}(A_W(\mathcal{Q}))$ for the output of $\mathcal{F}$. This notation suppresses the random seed s; to make this dependence explicit, we write $\mathcal{F}(s, A_W(\mathcal{Q}))$.

This abstract definition of witness finding algorithm is suitable for our information-theoretic lower bounds. Of course, we should consider more concrete algorithms for our upper bounds, including an appropriate definition of polynomial-time computability.

Definition 3. A witness finding algorithm $(\mathcal{Q}, \mathcal{F})$ is *polynomial-time* if the seed length $r(n)$ is bounded by a polynomial in n, and $\mathcal{Q}$ and $\mathcal{F}$ are polynomial-time algorithms taking $(1^n, s)$ as input where $s \in \{0,1\}^{r(n)}$; $\mathcal{Q}$ outputs a sequence of Boolean circuits $C_1, \ldots, C_m$ such that C_i computes a function $\{0,1\}^n \times \{0,1\}^{i-1} \to \{0,1\}$ with $Q_i = \{x \mid C_i(x, A_W(Q_1), \ldots, A_W(Q_{i-1})) = 1\}$, and $\mathcal{F}$ outputs a Boolean circuit computing $\mathcal{F}(A_W(\mathcal{Q}))$ as a function $\{0,1\}^m \to \{0,1\}^n$.

Remark 1. In many situations, a witness set W is determined by an "input" x to a witness finding algorithm. In the case of 3SAT, for example, x is a satisfiable 3CNF formula and W_x is the set of *its* satisfying assignments. However, it is not essential to consider such inputs in our abstract framework, so we omit specifying inputs to witness finding algorithms.

Remark 2. In actual witness finding algorithms for NP sets, we may relax some of the above conditions. For example, we may allow queries of the form "$|W \cap Q| \geq k$?" for some k that is polynomially bounded by n. The following discussion does not change much if such generalized queries are allowed. Our lower bound results essentially hold, and hence, we may not be able to improve our upper bound results by using such queries. Another relaxation is to allow an algorithm to output a list of polynomially many candidate witnesses; the algorithm *succeeds* if W contains any element in this list. We call an algorithm of this type a *witness-list finding algorithm*, and for any witness-list finding algorithm, we write $\ell(n)$ to denote its *list size*, a function bounding the number of elements in the list that the algorithm produces for any witness set in Ω_n. For the search version of an NP problem, this relaxation does not make any difference since one can check the correctness of a witness candidate in polynomial-time.

We now give a framework for discussing error tolerance.

Definition 4. For a witness finding algorithm $(\mathcal{Q}, \mathcal{F})$, we denote by $Dom \subseteq \{0,1\}^n$ the set of queries produced by $\mathcal{Q}$ for all possible random seeds and oracle answers. (For a polynomial-time algorithm, note that $|Dom|$ is bounded $2^{\mathrm{poly}(n)}$.) For a witness set W and distribution $\mathcal{D}$ on Dom, a function $\widetilde{A}_W : Dom \to \{0,1\}$ is *ε-error bounded* if its error probability is at most ε when queries are chosen uniformly from Dom; that is,

$$\Pr_{Q:Dom}[\ \widetilde{A}_W(Q) \neq A_W(Q)\ ] \ < \ \varepsilon$$

holds. For an algorithm $(\mathcal{Q}, \mathcal{F})$ with random seed length r, its *success probability with ε-error bounded oracle* is the probability $\Pr_{s:\{0,1\}^r}[\,\mathcal{F}(s, \widetilde{A}_W(\mathcal{Q}))$ succeeds$\,]$ with the worst ε-error bounded oracle for the worst witness set. An algorithm $(\mathcal{Q}, \mathcal{F})$ is *ε-error tolerant* it has $\Omega(1)$ success probability.

Note that the uniform distribution on Dom may be different from the distribution of queries made during the execution of the algorithm.

3 Witness Finding for Isolation

In this section we consider nonadaptive algorithms for the general witness finding problem. As mentioned in Section 1, these algorithms can be regarded as truth-table type witness isolation reductions.

We first state the algorithm of Ben-David et al. in our framework. Here we consider the error-free oracle A_W (we will consider error tolerance later in this section). See Example 1 in the Appendix for the explanation.

Proposition 1. There is a polynomial-time nonadaptive witness finding algorithm $(\mathcal{Q}_{\mathrm{BD}}, \mathcal{F}_{\mathrm{BD}})$ with $\Omega(1)$ success probability and $O(n^2)$ query complexity.

Our first result shows that $(\mathcal{Q}_{\mathrm{BD}}, \mathcal{F}_{\mathrm{BD}})$ is optimal in terms of query complexity.

Theorem 2. There is *no* nonadaptive witness finding algorithm with $\Omega(1)$ success probability and $o(n^2)$ query complexity.

Proof. Consider any nonadaptive witness finding algorithm $(\mathcal{Q}, \mathcal{F})$ which makes $m = o(n^2)$ queries. We will show that there exists a witness set W such that $\Pr_{s:\{0,1\}^r}[\,\mathcal{F}(s, A_W(\mathcal{Q})) \in W\,] = o(1)$. For our analysis, we use Yao's principle [Yao77]. That is, we have

$$\begin{aligned}
\min_{W} \Pr_{s:\{0,1\}^r}[\,\mathcal{F}(s, A_W(\mathcal{Q})) \in W\,] &\leq \max_{\rho} \min_{W} \Pr_{s:\rho}[\,\mathcal{F}(s, A_W(\mathcal{Q})) \in W\,] \\
&= \min_{\mathcal{W}} \max_{s\in\{0,1\}^r} \Pr_{W:\mathcal{W}}[\,\mathcal{F}(s, A_W(\mathcal{Q})) \in W\,] \\
&\leq \max_{s\in\{0,1\}^r} \Pr_{W:\mathcal{W}_*}[\,\mathcal{F}(s, A_W(\mathcal{Q})) \in W\,],
\end{aligned}$$

where W is any witness set, ρ is any distribution on $\{0,1\}^r$, $\mathcal{W}$ is any distribution on witness sets (i.e., on nonempty subsets of Ω), and $\mathcal{W}_*$ is a particular distribution on witness sets that we define below. Note that the lefthand term of the first inequality is what we would like to estimate. To prove the theorem, we need to show that $\Pr_{W:\mathcal{W}_*}[\mathcal{F}(s, A_W(\mathcal{Q})) \in W] = o(1)$ for any fixed $s \in \{0,1\}^r$.

Our distribution $\mathcal{W}_*$ on witness sets is defined by the following procedure to generate a witness set W: first we choose K uniformly at random from $[n]$, then we define W by including each $w \in \Omega$ in W independently with probability 2^{-K}. A small technicality is that W may possible be the empty set; however, since $W = \emptyset$ occurs with probability $o(1)$ for $W : \mathcal{W}_*$, we can ignore this degenerate case.[1] Note that, for $k \in [n]$, the expected size of W conditioned on $K = k$ is 2^{n-k}. Below we keep using K and W to denote these random variables.

We now fix an arbitrary seed $s \in \{0,1\}^r$. Let $Q_1, \ldots, Q_m$ be the queries given by $\mathcal{Q}(s)$, and let f be the function from $\{0,1\}^m$ to Ω given by $\mathcal{F}(s, \cdots)$ (which attempts to output a witness in W given the answers to queries $Q_1, \ldots, Q_m$). For $i \in [m]$, let $A_i = A_W(Q_i)$. That is, $A_i \in \{0,1\}$ is the indicator random variable for the event that $Q_i \cap W \neq \emptyset$.

Below by $\Pr[\cdots]$, we mean $\Pr_{W:\mathcal{W}_*}[\cdots]$. Then $\Pr[f(A_1, \ldots, A_m) \in W]$ is the success probability of the algorithm $(\mathcal{Q}, \mathcal{F})$ for the random seed s. Our goal is to show that $\Pr[f(A_1, \ldots, A_m) \in W] = o(1)$. Toward that end, we view $f(A_1, \ldots, A_m)$ as a random variable over Ω and estimate its entropy. Below we use the standard notations for discussing entropy.[2]

Claim 1. $\mathrm{H}(f(A_1, ..., A_m)) \;\leq\; \log n + O(m/n)$.

Proof. We have

$$\begin{aligned}
\mathrm{H}(f(A_1, ..., A_m)) &\leq \mathrm{H}(A_1, ..., A_m) \\
&\leq \mathrm{H}(A_1, ..., A_m, K) \;=\; \mathrm{H}(K) + \mathrm{H}(A_1, ..., A_m \mid K) \\
&\leq \log n + \sum_{i=1}^{m} \mathrm{H}(A_i \mid K) \;=\; \log n + \frac{1}{n} \sum_{i=1}^{m} \sum_{k=1}^{n} \mathrm{H}(A_i \mid K = k).
\end{aligned}$$

[1]Precisely speaking, we should consider a distribution $\mathcal{W}'_*$ conditioned that the empty set is never obtained. But the difference, which is negligible, is ignored here.

[2]For discrete random variables X and Y, $\mathrm{H}(X) = \sum_x -\Pr[X = x] \log \Pr[X = x]$ and $\mathrm{H}(X|Y) = \sum_y \Pr[Y = y]\mathrm{H}(X|Y = y)$, where $\mathrm{H}(X|Y = y) = \sum_x -\Pr[X = x|Y = y] \log \Pr[X = x|Y = y]$. For a sequence of random variables $X_1, \ldots, X_j$, $\mathrm{H}(X_1, \ldots, X_j)$ is the entropy of the joint distribution $(X_1, \ldots, X_j)$.

To finish the proof of the claim, we show that $\sum_{k=1}^{n} \mathrm{H}(A_i|K = k) = O(1)$ for all $i \in [m]$ (where $O(1)$ is some universal constant). Fix arbitrary $i \in [m]$. For $k \in [n]$, let $p_k = \Pr[A_i = 1|K = k]$ $(= \Pr[Q_i \cap W \neq \emptyset|K = k])$. We have $\mathrm{H}(A_i|K = k) = H(p_k)$ where $H : [0,1] \to [0,1]$ is the binary entropy function $H(p) = -p \log p - (1-p)\log(1-p)$. Note the inequality

$$\min(p, 1-p) \leq q \leq 1/2 \implies H(p) \leq -2q \log q.$$

We use this inequality to bound $H(p_k)$.

Let $\lambda = \log |Q_i|$. We consider three cases depending on $k \in [n]$:

- If $k \leq \lambda - 1$, then we have $1 - p_k = (1 - 2^{-k})^{|Q_i|} \leq \mathrm{e}^{-2^{\lambda-k}}$ $(< 1/2)$. Hence,

$$H(p_k) \leq -2\ln(\mathrm{e}^{-2^{\lambda-k}})\mathrm{e}^{-2^{\lambda-k}} \;=\; 2^{\lambda-k+1-\log(\mathrm{e})2^{\lambda-k}} \;=\; 2^{-\Omega(2^{\lambda-k})}.$$

- If $\lambda - 1 < k < \lambda + 1$, then $H(p_k) \leq 1$ (this case applies to at most two k's in $[n]$).
- If $k \geq \lambda + 1$, then $p_k \leq |Q_k|2^{-k} = 2^{\lambda-k}$ $(\leq 1/2)$. Hence, $H(p_k) \leq (k-\lambda)2^{\lambda-k+1}$.

Now it follows that

$$\begin{aligned}\sum_{k=1}^{n} \mathrm{H}(A_i \mid K = k) \;&<\; \sum_{k=1}^{n} H(p_k) \;=\; \sum_{k=1}^{\lfloor \lambda-1 \rfloor} 2^{-\Omega(2^{\lambda-k})} + 2 + \sum_{k=\lceil \lambda+1 \rceil}^{n} (k-\lambda)2^{\lambda-k+1} \\ &<\; O\left(\sum_{j=1}^{\infty} 2^{-\Omega(2^j)}\right) + 2 + O\left(\sum_{j=1}^{\infty} j2^{-j}\right) \;=\; O(1).\end{aligned}$$

□ Claim 1

Using this entropy bound, we now show that $\Pr[\, f(A_1, ..., A_m) \in W\,] = o(1)$. Fix an arbitrary constant $\epsilon > 0$. For $w \in \Omega$, let $p(w)$ denote the probability that $f(A_1, ..., A_m)$ takes value w. Let $U = \{w \mid p(w) \geq 2^{-\epsilon n}\}$ and note that $|U| \leq 2^{\epsilon n}$. For all $w \notin U$, we have $-\log p(w) > \epsilon n$ and thus (by Claim 1)

$$\begin{aligned}\Pr[\, f(A_1, ..., A_m) \notin U\,] \;&=\; \sum_{w \notin U} p(w) \;\leq\; \sum_{w \notin U} \frac{-p(w)\log p(w)}{\epsilon n} \\ &\leq\; \frac{\mathrm{H}(f(A_1, ..., A_m))}{\epsilon n} \;\leq\; \frac{\log n + O(m/n)}{\epsilon n} \;=\; o(1).\end{aligned}$$

Next we have

$$\begin{aligned}\Pr[\, U \cap W \neq \emptyset\,] \;&\leq\; \Pr[\, K \leq 2\epsilon n\,] + \Pr[\, U \cap W \neq \emptyset \mid K > 2\epsilon n\,] \\ &\leq\; 2\epsilon + \sum_{u \in U} \Pr[\, u \in W \mid K > 2\epsilon n\,] \\ &<\; 2\epsilon + |U|2^{-2\epsilon n} \;\leq\; 2\epsilon + 2^{-\epsilon n} \;=\; 2\epsilon + o(1).\end{aligned}$$

Combining these inequalities, we have

$$\begin{aligned}\Pr[\, f(A_1, ..., A_m) \in W\,] \;&\leq\; \Pr[\, f(A_1, ..., A_m) \notin U \,\vee\, U \cap W \neq \emptyset\,] \\ &\leq\; \Pr[\, f(A_1, ..., A_m) \notin U\,] + \Pr[\, U \cap W \neq \emptyset\,] \;\leq\; 2\epsilon + o(1).\end{aligned}$$

Since ϵ can be taken arbitrarily small, it follows that $\Pr[\, f(A_1, ..., A_m) \in W\,] = o(1)$. □

Next we consider error tolerance. Note that the algorithm $(\mathcal{Q}_{\mathrm{BD}}, \mathcal{F}_{\mathrm{BD}})$ is automatically $O(1/n^2)$-error tolerant since it makes only $O(n^2)$ queries. Here we present a new polynomial-time algorithm based on some standard error-correcting code that has a better $O(1/n)$-error tolerance while still making only $O(n^2)$ queries. (Note that in our framework we do not assume any error-free checking procedure for the obtained witnesses, as in [BT06]. One interesting point in the following algorithm is that we can indeed implement the checking procedure with erroneous oracles.)

Theorem 3. There is a polynomial-time, nonadaptive, and $O(1/n)$-error tolerant witness finding algorithm with $O(n^2)$ query complexity.

Proof. We first prepare some polynomial-time encodable and decodable code with some specific property suitable for our usage. Below for any binary strings x and y, we denote by $|x|$ the Hamming weight of x and by $\mathrm{dist}(x, y)$ the Hamming distance between x and y.

Lemma 4. There exists a polynomial-time encodable and decodable code with the following property for some constants $c > 1$ and $\delta > 0$: For any n, let $C \subseteq \{0,1\}^{2cn}$ denote the set of codewords encoding messages in $\{0,1\}^n$. Then we have (i) $|y| = cn$ for all $y \in C$, and for all distinct $y, y' \in C$, we have (ii) $\mathrm{dist}(y, y') \geq 2\delta cn$, and (iii) $|y \vee y'| = cn + \frac{1}{2}\mathrm{dist}(y, y') \geq (1+\delta)cn$, where $y \vee y'$ is the bit-wise OR of y and y'. Below we use C also to denote a function mapping any message in $\{0,1\}^n$ to the corresponding codeword in $\{0,1\}^{cn}$.

Proof. First consider any polynomial-time encodable and decodable binary code of some constant rate $1/c$ and constant relative minimum distance δ. That is, for each n, every message in $y \in \{0,1\}^n$ is encoded by some codeword $\widehat{y} \in \{0,1\}^{cn}$, and we have $\mathrm{dist}(\widehat{y}, \widehat{y}') \geq \delta cn$ for any two distinct codewords $\widehat{y}, \widehat{y}' \in \{0,1\}^{cn}$. For example, Justesen code satisfies all these requirements (see, e.g., [Rot06]). Now for any $x \in \{0,1\}^n$, our $C(x)$ is defined by

$$C(x) \; = \; (\widehat{y}_1, ..., \widehat{y}_{cn}, 1 - \widehat{y}_1, ..., 1 - \widehat{y}_{cn}),$$

where $\widehat{y} = \widehat{C}(x)$. It is easy to see that C satisfies properties (i), (ii) and (iii) of the lemma. □ Lemma 4

We present our algorithm $(\mathcal{Q}, \mathcal{F})$ as a modification of the algorithm of Ben-David et al. [BCGL92] by using this code. For this, we recall their algorithm first. The key tool of their algorithm is the isolation technique of Valiant and Vazirani [VV86]. In the abstract setting, their technique is a polynomial-time algorithm $\mathcal{Q}_{\mathrm{VV}}$ that generates/recognizes a random subset $\mathcal{Q}_{\mathrm{VV}}(s)$ of Ω $(= \{0,1\}^n)$ from a random seed $s \in \{0,1\}^{r_{\mathrm{iso}}}$ for some polynomial r_{iso} (again we write simply r_{iso} for $r_{\mathrm{iso}}(n)$); the important property here is that for any $W \subseteq \Omega$, we have $|\mathcal{Q}_{\mathrm{VV}}(s) \cap W| = 1$ with probability $\geq \frac{1}{4n}$ [VV86]. For $\mathcal{Q}_{\mathrm{VV}}(s)$ that *achieves isolation* (i.e., $|\mathcal{Q}_{\mathrm{VV}}(s) \cap W| = 1$), we can ask each bit of the unique witness $w \in \mathcal{Q}_{\mathrm{VV}}(s) \cap W$ nonadaptively. In our framework, this is implemented by asking queries $\{\, v \mid v \in \mathcal{Q}_{\mathrm{VV}}(s) \wedge v_i = 1 \,\}$ for all $i \in [n]$. The algorithm of Ben-David et al. asks a set of these n queries for $4n$ random seeds for s to achieve $\Omega(1)$ success probability. Clearly, in order for the algorithm to work, no error should occur for these $O(n^2)$ queries, and for this, we need an $O(1/n^2)$-error bounded oracle.

In our algorithm, instead of asking for each bit of the isolated witness directly, we ask for bits of the codeword of the isolated witness. That is, for each $s \in \{0,1\}^{r_{\rm iso}}$, we consider queries

$$Q_{s,j} \ = \ \{ \ v \mid v \in \mathcal{Q}_{\rm VV}(s) \ \wedge \ C(v)_j = 1 \ \}$$

for all $j \in [2cn]$. Again, to achieve $\Omega(1)$ success probability, the query algorithm $\mathcal{Q}$ asks a set of these $2cn$ queries for $4n$ independent random seeds $s_1, \ldots, s_{4n} \in \{0,1\}^{r_{\rm iso}}$. Thus, $\mathcal{Q}$ is an algorithm that takes random seeds $s_1, \ldots, s_{4n} \in \{0,1\}^{r_{\rm iso}}$, and asks queries $Q_{s_k,j}$ for all $k \in [4n]$ and $j \in [2cn]$. Hence, the algorithm makes $8cn^2$ nonadaptive queries. Let Dom denote the set $\{Q_{s,j} \mid s \in \{0,1\}^{r_{\rm iso}}, j \in [4cn]\}$ of all such queries.

For explaining the algorithm $\mathcal{F}$, we first see answers we can expect from an erroneous oracle. Let $\widetilde{A}_W : 2^\Omega \to \{0,1\}$ be any $\frac{\delta}{32n}$-error bound oracle for some unknown witness set W. That is, we assume that

$$\Pr_{Q:Dom}[\widetilde{A}_W(Q) \neq A_W(Q)] \ \ (= \Pr_{s:\{0,1\}^r, j:[2cn]}[\widetilde{A}_W(Q_{s,j}) \neq A_W(Q_{s,j})]) \ \le \ \frac{\delta}{32n}.$$

For $s \in \{0,1\}^{r_{\rm iso}}$, we define

$$\alpha(s) \ = \ \frac{1}{2cn} \sum_{j \in [2cn]} A_W(Q_{s,j}) \quad \text{ and } \quad \widetilde{\alpha}(s) \ = \ \frac{1}{2cn} \sum_{j \in [2cn]} \widetilde{A}_W(Q_{s,j}).$$

Let $\mathsf{Good}(s)$ be the event that $|\alpha(s) - \widetilde{\alpha}(s)| < \delta/4$. By Markov's inequality we have

$$\begin{aligned} \Pr_s[\neg\mathsf{Good}(s)] \ &= \ \Pr_s[\Pr_j[\widetilde{A}_W(Q_{s,j}) \neq A_W(Q_{s,j})] \ \ge \ \delta/4] \\ &\le \ \frac{\mathrm{E}_s[\Pr_j[\widetilde{A}_W(Q_{s,j}) \neq A_W(Q_{s,j})]]}{\delta/4} \\ &= \ \frac{\Pr_{s,j}[\widetilde{A}_W(Q_{s,j}) \neq A_W(Q_{s,j})]}{\delta/4} \ \le \ \frac{1}{8n}. \end{aligned} \tag{1}$$

Let $\mathsf{Isolated}(s)$ be the event that $|\mathcal{Q}_{\rm VV}(s) \cap W| = 1$.

Claim 2. If $\mathsf{Good}(s)$ holds, then $\mathsf{Isolated}(s) \Leftrightarrow |\frac{1}{2} - \widetilde{\alpha}(s)| < \frac{\delta}{4}$. Moreover, if $\mathsf{Good}(s) \wedge \mathsf{Isolated}(s)$ holds, then given $\widetilde{A}_W(Q_{s,1}), \ldots, \widetilde{A}_W(Q_{s,2cn})$, the unique element in $\mathcal{Q}_{\rm VV}(s) \cap W$ can be computed in polynomial-time.

Proof. The claim follows from the properties of C. If $\mathsf{Good}(s)$, then

$$\begin{aligned} |\mathcal{Q}_{\rm VV}(s) \cap W| \ = \ 0 \ &\Rightarrow \ \alpha(s) \ = \ 0 && \Rightarrow \ \widetilde{\alpha}(s) \ < \ \frac{\delta}{4} \left(< \frac{1}{2} - \frac{\delta}{4} \right), \\ |\mathcal{Q}_{\rm VV}(s) \cap W| \ = \ 1 \ &\Rightarrow \ \alpha(s) \ = \ \frac{1}{2} && \Rightarrow \ \frac{1}{2} - \frac{\delta}{4} \ < \ \widetilde{\alpha}(s) \ < \ \frac{1}{2} + \frac{\delta}{4}, \\ |\mathcal{Q}_{\rm VV}(s) \cap W| \ \ge \ 2 \ &\Rightarrow \ \alpha(s) \ \ge \ \frac{1}{2} + \frac{\delta}{2} && \Rightarrow \ \widetilde{\alpha}(s) \ > \ \frac{1}{2} + \frac{\delta}{4}. \end{aligned}$$

Thus, $\mathsf{Good}(s)$ implies $\mathsf{Isolated}(s) \Leftrightarrow |\frac{1}{2} - \widetilde{\alpha}(s)| < \frac{\delta}{4}$. In the event that both $\mathsf{Good}(s)$ and $\mathsf{Isolated}(s)$ hold, we can decode $(\widetilde{A}_W(Q_{s,1}), ..., \widetilde{A}_W(Q_{s,2cn})) \in \{0,1\}^{2cn}$ in polynomial-time to recover the original message $w \in \{0,1\}^n$; this w is then the unique element of $\mathcal{Q}_{\rm VV}(s) \cap W$. □ Claim 2

Now the following description of algorithm $\mathcal{F}$ is clear from this claim: For given answers from the oracle, compute $\widetilde{\alpha}(s_i)$ for all $i \in [4n]$. If there exists $i \in [4n]$ such that $|\frac{1}{2} - \widetilde{\alpha}(s_i)| < \frac{\delta}{4}$, then for the first such i, output the unique element of $\mathcal{Q}_{\mathrm{VV}}(s_i) \cap W$ according to Claim 2 under the (possibly false) assumption that $\mathsf{Good}(s_i)$ holds. (If $|\frac{1}{2} - \widetilde{\alpha}(s_i)| \geq \frac{\delta}{4}$ for all $i \in [4n]$, the algorithm simply fails.)

To analyze its success probability, note that it successfully outputs a witness in W whenever $\bigwedge_{i\in[4n]} \mathsf{Good}(s_i) \wedge \bigvee_{i\in[4cn]} \mathsf{Isolated}(s_i)$ holds. Using the bound (1) and the fact that $\Pr_{s:\{0,1\}^{r_{\mathrm{iso}}}}[\mathsf{Isolated}(s)] \geq \frac{1}{4n}$, we have

$$\begin{aligned}
&\Pr_{s_1,\ldots,s_{4n}}\left[\ \bigwedge_{i\in[4n]} \mathsf{Good}(s_i) \wedge \bigvee_{i\in[4n]} \mathsf{Isolated}(s_i)\ \right]\\
&\geq\ 1 - \Pr_{s_1,\ldots,s_{4n}}\left[\bigvee_{i\in[4n]} \neg\mathsf{Good}(s_i)\right] - \Pr_{s_1,\ldots,s_{4n}}\left[\bigwedge_{i\in[4n]} \neg\mathsf{Isolated}(s_i)\right]\\
&\geq\ 1 - 4n\Pr_s[\neg\mathsf{Good}(s)] - \Big(1 - \Pr_s[\neg\mathsf{Isolated}(s)]\Big)^{4n}\ \geq\ 1 - \frac{1}{2} - \frac{1}{\mathrm{e}}\ >\ 0.13.
\end{aligned}$$

Therefore, the algorithm succeeds with constant probability > 0.13 for any $\frac{\delta}{32n}$-error bounded oracle. $\square$

4 Witness Finding for Decoding

We consider the case where a witness set is restricted to a singleton set, the situation typical for decoding problems. Throughout this section, we consider only singleton witness sets, and the conditions of, e.g., Definition 2 are modified for this restriction. Also, since our target witness set is singleton, we specify the target by a witness $w \in \Omega$ instead of a witness set $W = \{w\}$ in the following. Throughout this section, we use ε to denote any function on $\mathbf{N}$ such that $0 < \varepsilon(n) < 1/2$ holds for any n.

First we state the algorithm of Goldreich-Levin in our framework.

Proposition 5. There is a polynomial-time, nonadaptive, and $(1/2-\varepsilon)$-error tolerant witness-list finding algorithm $(\mathcal{Q}_{\mathrm{GL}}, \mathcal{F}_{\mathrm{GL}})$ with $O(n^2/\varepsilon^2)$ query complexity and $O(n/\varepsilon^2)$ list size.

Let us call the parameter ε above *advantage*. The algorithm of Goldreich-Levin achieves $n^{-O(1)}$ advantage within polynomially bounded query complexity and list size. As we will see below, $n^{-\omega(1)}$ advantage is impossible within polynomially bounded query complexity and list size, and in this sense, the Goldreich-Levin algorithm is optimal. On the other hand, the tradeoff between advantage and query complexity can be improved as follows. It should be noted, however, that the algorithm given below is not polynomial-time as it is, and showing a similar relation by some polynomial-time algorithm is open.

Theorem 6. There is a *deterministic*, nonadaptive, and $(1/2-\varepsilon)$-error tolerant witness-list finding algorithm with $O(n/\varepsilon^2)$ query complexity and $O(1/\varepsilon^2)$ list size.

Proof. The proof is almost trivial from the following lemma, which provides an optimal list-decodable code.

Lemma 7. ([Eli91, GHSZ02]) For some $m = O(n/\varepsilon^2)$, there exists a set $C \subseteq \{0,1\}^m$ of size 2^n such that for every $v \in C$, we have $|B(1/2 - \varepsilon, v) \cap C| = O(1/\varepsilon^2)$, where $B(\gamma, v) \subseteq \{0,1\}^m$ denotes the Hamming ball of radius γm centered at v.

We use this lemma to design our algorithm. Consider any ε and fix any n. Let C be the set given in Lemma 7. We associate every witness $w \in \Omega$ $(= \{0,1\}^n)$ with an element in C in any one-to-one manner. Denote by $C(w)$ an element in C corresponding to $w \in \Omega$. Our algorithm is deterministic and works as follows: The i-th query is given by $Q_i = \{v : C(v)_i = 1\}$, where $C(v)_i$ denotes the ith bit of $C(v) \in \{0,1\}^m$. Namely, the query "$\{w\} \cap Q_i \neq \emptyset$?" asks whether the i-th bit of $C(w)$ is 1 or not. On the answers $\widetilde{A}_w(Q_1), ..., \widetilde{A}_w(Q_m)$, the algorithm outputs a list of all the elements of $\{v | C(v) \in B(1/2 - \varepsilon, \widetilde{\alpha}_w) \cap C\}$, where $\widetilde{\alpha}_w := (\widetilde{A}_w(Q_1), ..., \widetilde{A}_w(Q_m)) \in \{0,1\}^m$. Note that we do not know an efficient way for enumerating these elements and thus the algorithm makes such a list by brute force.

From a coding-theoretic viewpoint, w is an n-bit message, $C(w)$ is its codeword of m-bit length, and $\widetilde{\alpha}_w$ is an corrupted codeword. Hence, if $\widetilde{A}_w$ makes at most $(1/2 - \varepsilon)m$ errors, then we have the target witness in $\{v | C(v) \in B(1/2 - \varepsilon, \widetilde{\alpha}_w) \cap C\}$. Clearly, the algorithm makes m queries and it follows from Lemma 7 that the list size is $O(1/\varepsilon^2)$. □

We show the tightness of this tradeoff in the next theorem. In fact, if we require both constant success probability and polynomially bounded list size, then the next theorem shows that $m > c_1 n/\varepsilon^2$ queries are necessary while the above theorem shows that $m = \widehat{c}_1 n/\varepsilon^2$ is sufficient for some constants $c_1 < \widehat{c}_1$.

Theorem 8. Consider any witness-list finding algorithm with seed length r, query complexity m, and list size ℓ. For any sufficiently small $\varepsilon > 0$, let α denote its success probability with $(1/2 - \varepsilon)$-error bounded oracle. Then for some constants $c_1 > 0$, if $m \leq c_1 n/\varepsilon^2$, then we have $\alpha = O(\ell 2^{-\Omega(n)}/\varepsilon)$. (In the proof below, we use $c_1 = 1/640$.)

Proof. Consider any witness-list finding algorithm $(\mathcal{Q}, \mathcal{F})$ with random seed length r, query complexity m, and list size ℓ. Also consider any $\varepsilon > 0$ for the advantage parameter. We may assume that $\varepsilon < 1/4$. As usual, fix sufficiently large n, and consider the problem of finding a given witness $w \in \Omega = \{0,1\}^n$; from now on, r, m, ℓ, and ε are some numbers determined by n. In particular, let us fix the query number bound m to $m = c_1 n/\varepsilon^2$ for $c_1 = 1/640$.

Let *Dom* be the set of all possible queries made by $\mathcal{Q}$ with any random seed and any oracle answers. We consider two cases depending on $D := |Dom|$; the case (a) where $D \leq d_1 \varepsilon^{-2}$ for some sufficiently large constant d_1 (that will be specified later), and the case (b) where $D > d_1 \varepsilon^{-2}$.

Case (a): We list all possible queries in *Dom* under some ordering, and let $Q_1, ..., Q_D$ be this list. Oracle answers to these queries can be specified naturally by a string $a \in \{0,1\}^D$. For any $a \in \{0,1\}^D$ and $w \in \Omega$, we say that a and w are $(1/2+\varepsilon)$-*consistent* if the Hamming distance between a and $A_w(Q_1)A_w(Q_2)\cdots A_w(Q_D)$ is $\leq D' := (1/2 - \varepsilon)D$. Note that each $w \in \Omega$ has $\binom{D}{D'}$ strings that are $(1/2+\varepsilon)$-consistent with w; hence, the total number of pairs (a, w) that are $(1/2+\varepsilon)$-consistent is at least $2^n\binom{D}{D'}$. Thus, there exists some $a_0 \in \{0,1\}^D$ that is $(1/2+\varepsilon)$-consistent with at least

$$\frac{2^n\binom{D}{D'}}{2^D} \;\geq\; \frac{2^n c_{\rm str}2^D}{2^D\sqrt{D}\exp(4\varepsilon^2 D)} \;\geq\; \frac{c_{\rm str}2^n}{\sqrt{D}\exp(4d_1)} \;=\; O\left(\varepsilon 2^n\right)$$

witnesses, where we use the bound given by Claim 3 below for $\binom{D}{D'}$. Let W_0 denote the set of such witnesses.

Now we again follow Yao's principle and consider some distribution on witnesses while we fix a random seed used by $(\mathcal{Q},\mathcal{F})$ to any s in $\{0,1\}^r$. Our distribution is simply the uniform distribution on W_0. Note that a_0 is $(1/2-\varepsilon)$-error bounded for any witness $w\in W_0$. Thus, for any $w\in W_0$, we consider the execution of the algorithm with random seed s and oracle answers specified by a_0. Then the output, i.e., the list of ℓ candidates for w, is fixed, and it can be correct for at most ℓ witnesses $w\in W_0$. Hence, under the uniform distribution on W_0, the success probability of our algorithm (with respect to s and a_0) is at most $\frac{\ell}{W_0}=O\left(\ell 2^{-n}/\varepsilon\right)$, which can be used as an upper bound for the success probability of the algorithm in the worst case by Yao's principle.

Case (b): Consider the case (b), i.e., the case where *Dom* is sufficiently large. Let $\mathcal{OK}_w$ denote the set of all oracles that are $(1/2-\varepsilon)$-error bounded. Our goal is to estimate the success probability α defined by

$$\alpha \;=\; \min_{w\in\Omega}\;\min_{\widetilde{A}_w\in\mathcal{OK}_w}\;\Pr_{s:\{0,1\}^r}[\;\mathcal{F}(s,\widetilde{A}_w(\mathcal{Q}))\ni w\;].$$

Again by using Yao's principle, we consider some distributions on $w\in\Omega$ and on $\widetilde{A}_w\in\mathcal{OK}_w$, and discuss the probability of $\mathcal{F}(s,\widetilde{A}_w(\mathcal{Q}))\ni w$ under these distributions while any $s\in\{0,1\}^r$ is fixed.

More precisely, we will bound α by

$$\begin{aligned}\alpha &= \min_{w\in\Omega}\;\min_{\widetilde{A}_w\in\mathcal{OK}_w}\;\Pr_{s:\{0,1\}^r}[\;\mathcal{F}(s,\widetilde{A}_w)=w\;] \;\leq\; \max_{\rho}\min_{w\in\Omega}\;\min_{\widetilde{A}_w\in\mathcal{OK}_w}\;\Pr_{s:\rho}[\;\mathcal{F}(s,\widetilde{A}_w)=w\;]\\ &= \min_{\mu_1}\min_{\mu_2}\;\max_{s\in\{0,1\}^r}\;\Pr_{w:\mu_1,\widetilde{A}_w:\mu_2}[\;\mathcal{F}(s,\widetilde{A}_w)=w\;] \;\leq\; \max_{s\in\{0,1\}^r}\;\Pr_{w:\Omega',\widetilde{A}_w:\mathcal{OK}'_w}[\;\mathcal{F}(s,\widetilde{A}_w)=w\;],\end{aligned}$$

where ρ, μ_1, and μ_2 are distributions on $\{0,1\}^r$, Ω, and $\mathcal{OK}_w$ respectively, and we use Ω' and $\mathcal{OK}'_w$ to denote also some distributions on Ω and $\mathcal{OK}_w$ that are defined below.

For the distribution of witnesses, we consider the uniform distribution on Ω; the symbol Ω is used also for denoting this distribution. For defining a distribution on $\mathcal{OK}_w$ and for analyzing the adaptive query computation of the algorithm, we use a folklore argument attributed to Rudich (see, e.g., [GNW95]). We use, as a "noise function", a random function Δ defined on *Dom*; Δ takes, for each $Q\in$ *Dom* independently, value 0 with probability $1/2+2\varepsilon$ and value 1 with probability $1/2-2\varepsilon$. Then we assume that our oracle is generated by $\widetilde{A}_w(Q)=A_w(Q)\oplus\Delta(Q)$; this is our distribution[3] on $\mathcal{OK}_w$, which is again denoted by $\mathcal{OK}_w$.

[3]Note that under this distribution some $\widetilde{A}_w$ may not be $(1/2-\varepsilon)$-error bounded although this probability is small since $|Dom|\geq d_1\varepsilon^{-2}$ and d_1 is sufficiently large. Thus, precisely speaking, the distribution that we should consider is the one that defines oracles by using Δ satisfying the condition that $|\{Q\in Dom|\Delta(Q)=1\}|\leq(1/2-\varepsilon)|Dom|$. But the difference is within at most constant factor, and we argue here by using the distribution $\mathcal{OK}_w$ defined above.

Now we fix an arbitrary random seed $s \in \{0,1\}^r$. Since s is fixed, the computation of the algorithm and the produced list of witness candidates are determined by a sequence of answers from oracle $\widetilde{A}_w$ that is determined by w and Δ. We use a string $a \in \{0,1\}^m$ to denote this *answer sequence* that can be also regarded as a *computation path* of the algorithm. (If the number of queries is smaller than m on some path, then a prefix a' of a is used to determine the computation. That is, in this case, we consider that $a'u$ yields the same answer for any $u \in \{0,1\}^{m-|a'|}$; for the sake of the following analysis, we regard the case $u_i = 1$ as the case that the oracle makes an error.) Note that, no matter which w is given, every answer sequence a occurs depending on Δ, and that the algorithm behaves in the same way on the same answer sequence. On the other hand, depending on w, the probability that each a occurs may differ. For each $w \in \Omega$ and $a \in \{0,1\}^m$, let $E_{a,w}$ denote the event that the algorithm receives this answer sequence a with respect to w, and let $\#_1(a,w)$ denote the number of queries Q such that $\Delta(Q) = 1$. Then the probability $p(a,w)$ that this event holds is

$$p(a,w) \;=\; \left(\frac{1}{2} - 2\varepsilon\right)^{\#_1(a,w)} \left(\frac{1}{2} + 2\varepsilon\right)^{m-\#_1(a,w)}. \tag{2}$$

Here we assume, without loss of generality, that all queries on each computation path are different and hence errors occur independently along each computation path.

We fix any $w \in \Omega$ and discuss the algorithm's success probability α_w. We say that an answer sequence is *good* (with respect to w) if it yields a list containing w. Though $E_{a,w}$ and $E_{a',w}$ may be correlated, they are disjoint; hence, we have

$$\alpha_w \;=\; \sum_{a:\text{good}} p(a,w) \;=\; \sum_{a:\text{good}} \left(\frac{1}{2} - 2\varepsilon\right)^{\#_1(a,w)} \left(\frac{1}{2} + 2\varepsilon\right)^{m-\#_1(a,w)}.$$

We would like to express the number M_w of good answer sequences in terms of this α_w. Consider the ordering of sequences a with respect to $p(a,w)$. Note that M_w is minimized if the set of good sequences consists of the first M_w sequences in this ordering. For the case where $\alpha_w < 1/2$, let δ_w be the smallest number satisfying

$$\alpha_w \;\le\; \sum_{i=0}^{k_w} \binom{m}{i} \left(\frac{1}{2} - 2\varepsilon\right)^{i} \left(\frac{1}{2} + 2\varepsilon\right)^{m-i}, \tag{3}$$

where $k_w = \lfloor (1/2 - 2\varepsilon - \delta_w)m \rfloor$. Then we have $\delta_w > 0$, and by the Hoeffding bound the righthand side is bounded as

$$\sum_{i=0}^{k_w} \binom{m}{i} \left(\frac{1}{2} - 2\varepsilon\right)^{i} \left(\frac{1}{2} + 2\varepsilon\right)^{m-i} \;\le\; \frac{1}{2}\exp(-2\delta_w^2 m).$$

Hence,

$$\alpha_w \;\le\; \frac{1}{2}\exp(-2\delta_w^2 m). \tag{4}$$

On the other hand, for the case where $\alpha_w \ge 1/2$, we simply set $k_w = \lfloor (1/2 - 2\varepsilon)m \rfloor$. Then from the above observation (by letting $\delta_w = 0$ if $\alpha_w \ge 1/2$), we have

$$M_w \;\ge\; \sum_{i=0}^{k_w - 1} \binom{m}{i} \;\ge\; \binom{m}{k_w} \;\ge\; \frac{c_{\text{str}} 2^m}{\sqrt{m}} \exp(-4(2\varepsilon + \delta_w)^2 m), \tag{5}$$

where the last bound is from Claim 3 below.

Now we bound M_w by α_w. First consider the special case where $0 < \varepsilon \leq \delta_w$. In this case, we have $\alpha_w < 1/2$, and from (4), we immediately have $2\alpha_w \leq \exp(-2\delta_w^2 m) \leq \exp(-2\varepsilon^2 m) \leq \exp(-2c_1 n) = 2^{\Omega(-n)}$, which leads to our desired bound. Thus, in the following, we consider the case $\delta_w \leq \varepsilon$. Recall that $m = c_1 n/\varepsilon^2$ for $c_1 = 1/640$. Then from (4) and (5) (and using the bound $\alpha_w \leq 1$ for the case $\alpha_w \geq 1/2$), we have, for some constant $d_2 > 0$, that

$$\begin{aligned}
M_w &\geq \frac{c_{\text{str}} 2^m}{\sqrt{m}} \exp(-4(2\varepsilon + \delta_w)^2 m) \\
&= \frac{c_{\text{str}} 2^m}{\sqrt{m}} \exp(-16\varepsilon^2 m) \exp(-16\varepsilon\delta_w m) \exp(-4\delta_w^2 m) \\
&\geq \frac{c_{\text{str}} 2^m}{\sqrt{m}} \exp(-32\varepsilon^2 m) \exp(-4\delta_w^2 m) \;\geq\; \frac{c_{\text{str}} 2^m}{\sqrt{m}} \exp(-32c_1 n)(2\alpha_w)^2/4 \\
&\geq \frac{d_2 2^m 2^{-n/10}}{\sqrt{m}} \alpha_w^2, \qquad (6)
\end{aligned}$$

where the last bound follows from our choice of $c_1 = 1/640$.

Note that each answer sequence cannot be good for more than ℓ witnesses, and that there are 2^m answer sequences. Hence, we have $\sum_{w\in\Omega} M_w \leq \ell 2^m$, and thus from (6) we have $\sum_{w\in\Omega} \alpha_w^2 \leq \frac{\ell\sqrt{m}2^{n/10}}{d_2}$. Then by the Cauchy-Schwartz inequality, we have

$$\left(\sum_{w\in\Omega} \alpha_w\right)^2 \leq \left(\sum_{w\in\Omega} \alpha_w^2\right) \cdot \left(\sum_{w\in\Omega} 1\right) \leq \frac{\ell\sqrt{m}2^{n/10}2^n}{d_2} = \frac{\ell\sqrt{m}2^{11n/10}}{d_2}.$$

Thus, we have

$$\begin{aligned}
\Pr_{w:\Omega,\, \widetilde{A}_w:\mathcal{OK}_w} [\mathcal{F}(s, \widetilde{A}_w) = w] &= \frac{1}{2^n} \sum_{w\in\Omega} \alpha_w \;\leq\; \frac{\sqrt{\ell\sqrt{m}}2^{-9n/20}}{\sqrt{d_2}} \\
&= O\left(\left(\ell\varepsilon^{-1}\right)^{1/2} 2^{-\Omega(n)}\right) = O\left(\frac{\ell 2^{-\Omega(n)}}{\varepsilon}\right).
\end{aligned}$$

Since this holds for all $s \in \{0,1\}^r$, by Yao's principle, we have the desired bound for α. □

Claim 3. By the Stirling bound, for some constant $c_{\text{str}} > 0$ and for any $\epsilon \leq 1/4$, we have

$$\binom{u}{(\frac{1}{2} - \epsilon)u} \geq \frac{c_{\text{str}} 2^u}{\sqrt{u}} \exp(-4\epsilon^2 u).$$

Proof. We use the following approximation by Stirling: For some constants c_{low} and c_{up} and for any $n \geq 1$, we have

$$c_{\text{low}}\sqrt{n}\left(\frac{n}{\mathrm{e}}\right)^n < n! < c_{\text{up}}\sqrt{n}\left(\frac{n}{\mathrm{e}}\right)^n.$$

Let $\beta = 1/2 - \epsilon$. Then for some constant $d > 0$, we have

$$\begin{aligned}
\binom{u}{\beta u} &= \frac{u!}{((1-\beta)u)!(\beta u)!} \geq \frac{d}{\sqrt{\beta(1-\beta)u}} \cdot \frac{u^u}{((1-\beta)u)^{(1-\beta)u}(\beta u)^{\beta u}} \\
&= \frac{d}{\sqrt{\beta(1-\beta)u}} \cdot (1-\beta)^{-(1-\beta)u}\beta^{-\beta u}.
\end{aligned}$$

This can be modified further to

$$\begin{aligned}
\binom{u}{\beta u} &\geq \frac{d}{\sqrt{(1/4-\epsilon^2)u}} \cdot \left(\frac{1}{2}-\epsilon\right)^{-(1/2-\epsilon)u} \left(\frac{1}{2}+\epsilon\right)^{-(1/2+\epsilon)u} \\
&= \frac{d}{\sqrt{(1/4-\epsilon^2)u}} \cdot \left(\frac{1}{4}-\epsilon^2\right)^{-u/2+\epsilon u} \left(\frac{1}{2}+\epsilon\right)^{-2\epsilon u} \\
&\geq \frac{d}{\sqrt{(1/4-\epsilon^2)u}} \cdot 4^{u/2-\epsilon u} 2^{2\epsilon u} (1+2\epsilon)^{-2\epsilon u} \\
&= \frac{d 2^u}{\sqrt{(1/4-\epsilon^2)u}} \cdot (1+2\epsilon)^{-2\epsilon u} \;\geq\; \frac{c_{\mathrm{str}} 2^u}{\sqrt{u}} \cdot \exp(-4\epsilon^2 u).
\end{aligned}$$

□

Recall that the algorithm given in Theorem 6 outputs a list of size $O(1/\varepsilon^2)$ even with the optimal query complexity. In fact, this list size is optimal up to a constant factor subject to achieving constant success probability. More precisely, we have the following bound.

Theorem 9. Consider any witness-list finding algorithm with list size ℓ, and for any sufficiently small $\varepsilon > 0$, let α be its success probability with $(1/2-\varepsilon)$-error bounded oracle. Then for some constants $c_2 > 0$, if $n > c_2 \log(1/\varepsilon)/\varepsilon^2$, then we have $\alpha = O(\ell\varepsilon^2)$.

Proof. The proof follows immediately from the lemma stated below, which is a special case of the result of Guruswami and Vadhan [GV10] on a lower bound of list size for list-decoding, and the outline of Case (a) in the proof of Theorem 8 with Yao's principle.

Lemma 10. ([GV10]) For some constants $d_1, d_2 > 0$, the following holds: For any integer $D > 0$ and any $C \subseteq \{0,1\}^D$, if $\varepsilon < d_1$ and $|C| > 2^{d_2/2\varepsilon^2 \log(1/\varepsilon)}$, then there exists some $a_0 \in \{0,1\}^D$ such that $|C \cap B(1/2-\varepsilon, a_0)| = \Omega(\varepsilon^{-2})$.

Consider any witness-list finding algorithm $(\mathcal{Q}, \mathcal{F})$. Consider any advantage ε, $0 < \varepsilon < d_1$, and let c_2 be the constant d_2 specified in Lemma 10. Let $D = |Dom|$, and we list all possible queries $Q_1, ..., Q_D$ in Dom under some ordering. We view C in Lemma 10 as the set of oracle answers $A_w(Q_1), ..., A_w(Q_D)$ to these queries for some $w \in \Omega$ $(= \{0,1\}^n)$. We may assume that $|C| \geq 2^{n/2}$; since otherwise, there are large number of w's with the same oracle answers, and the success probability α becomes small and the desired upper bound for α can be shown rather easily (by using the condition that $n > c_2/\varepsilon^2 \log(1/\varepsilon)$). Thus, the condition $|C| > 2^{d_2/2\varepsilon^2 \log(1/\varepsilon)}$ of the lemma is satisfied if $n > c_2/\varepsilon^2 \log(1/\varepsilon)$. Then by Lemma 10, there exists $a_0 \in \{0,1\}^D$ that is $(1/2+\varepsilon)$-consistent with $\Omega(\varepsilon^{-2})$ witnesses. Let W_0 be the set of such witnesses. Now, we again follow Yao's principle, and by the same reasoning of Case (a) in the proof of Theorem 8, the desired upper bound $\ell/|W_0| = O(\ell\varepsilon^2)$ for α is derived. □

Acknowledgements. We thank Valentine Kabanets for fruitful discussions. AK was supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B) No.21300002.

References

[AB09] S. Arora and B. Barak, *A Computational Complexity: A Modern Approach*, Cambridge Univ. Press, 2009.

[BCGL92] S. Ben-David, B. Chor, O. Goldreich, and M. Luby, On the theory of average-case complexity, *Journal of Computer and System Sciences*, 44(2):193–219, 1992.

[BT06] A. Bogdanov and L. Trevisan, On worst-case to average-case reductions for NP problems, *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[Eli91] P. Elias, Error-correcting codes for list decoding, *IEEE Transactions on Information Theory*, 37(1):5–12, 1991.

[Gol01] O. Goldreich, *Foundations of Cryptography, Basic Tools*, Cambridge Univ. Press, 2001.

[GL89] O. Goldreich and L. Levin, Hard-core predicates for any one-way function, in *Proc. 21st ACM Sympos. on the Theory of Comput.*, 25–32, 1989.

[GNW95] O. Goldreich, N. Nisan, and A. Wigderson, On Yao's XOR-lemma, Technical Report TR95-050, ECCC Report, 1995.

[GHSZ02] V. Guruswami, J. Håstad, M. Sudan, and D. Zuckerman, Combinatorial bounds for list decoding, *IEEE Transactions on Information Theory*, 48(5):1021–1035, 2002.

[GV10] V. Guruswami and S. Vadhan, A lower bound on list size for list decoding, *IEEE Transactions on Information Theory*, 56(11):5681–5688, 2010.

[KW11] V. Kabanets and O. Watanabe, Is the Valiant-Vazirani isolation lemma improvable?, Technical Report TR11-151, ECCC Report, 2011.

[MVV87] K. Mulmuley, U. Vazirani, and V. Vazirani, Matching is as easy as matrix inversion, *Combinatorica*, 7(1):105–113, 1987.

[Rot06] R.M. Roth, *Introduction to Coding Theory.* Cambridge Univ. Press, 2006.

[VV86] L. Valiant and V. Vazirani, NP is as easy as detecting unique solutions, *Theoretical Computer Science*, 47:85–93, 1986.

[Yao77] A.C. Yao, Probabilistic computations: toward a unified measure of complexity, *Proc. of the 18th IEEE Sympos. on Foundations of Comput. Sci.*, IEEE, 222–227, 1977.

Appendix: Related Work

We state algorithms studied in related work by using our framework, thereby showing how they are related to our investigation.

Example 1. (Islolation Technique and the Algorithm of Ben-David et al.)

Valiant and Vazirani [VV86] gave a reduction $f_{\rm iso}$ from, e.g., 3SAT to some NP problem $L_{\rm iso}$, which has been called an "isolation technique" or an "isolation reduction." The reduction $f_{\rm iso}$ is a randomized function, and for any $\phi \in$ 3SAT with n variables, it yields an instance $f_{\rm iso}(\phi)$ for the problem $L_{\rm iso}$ that has the following properties with reasonable probability: (1) $f_{\rm iso}(\phi) \in L_{\rm iso}$, and (2) it is witnessed (for $L_{\rm iso}$'s witness system) by a *unique* witness $w \in \{0,1\}^n$ that is indeed one of the satisfying assignments of ϕ. We say that $f_{\rm iso}(\phi)$ succeeds the isolation if both (1) and (2) hold. It is shown that $f_{\rm iso}(\phi)$ succeeds the isolation with probability $\Omega(1/n)$ for any $\phi \in$ 3SAT with n variables. They used this reduction to show that a (promised) UniqueSAT problem is hard for NP. Since then the technique has been used as a key tool in computational complexity theory.

Ben-David et al. [BCGL92] used this technique to define a randomized algorithm solving an NP-type search problem by using some NP set as an oracle. Here we state one variation of their algorithm for the problem of searching a satisfying assignment of 3SAT instances. For any n, and let 3SAT$_n$ denote the set of 3SAT formulas with n variables. Consider any $\phi \in$ 3SAT$_n$; its satisfying assignments are strings in $\{0,1\}^n$, and let W_ϕ denote the set of all satisfying assignments of ϕ. Their algorithm asks $O(n^2)$ queries to some oracle set D in NP, which consists of $O(n)$ sets of the following queries: $(y,i,0)$ and $(y,i,1)$ for all $i \in [n]$, where $y = f_{\rm iso}(\phi)$. For each query (y,i,b), the NP oracle answers 'yes' iff there is some witness of $L_{\rm iso}$ for y such that its i-th bit is b. That is, our oracle set D is defined by

$$D \; = \; \{ \; (y,i,b) \mid \exists w \, [\, w \text{ is a witness for } y \in L_{\rm iso} \text{ and } w_i = b \,] \; \},$$

where by "w_i" we mean the i-th bit of w. It is easy to see that D is in NP. Furthermore, if $y = f_{\rm iso}(\phi)$ succeeds the isolation, then y has a unique witness w in W_ϕ and exactly one of $(y,i,0)$ or $(y,i,1)$ gets 'yes' answer for all $i \in [n]$; hence, in this case, this w can be constructed from these answers, and one can be sure that w is the correct satisfying assignment of ϕ. Since the isolation succeeds with probability $\Omega(1/n)$, by asking the above set of queries for $c_{\rm iso}n$ independent values of $f_{\rm iso}(\phi)$ (for some constant $c_{\rm iso}$) and then by choosing one set of queries where the isolation succeeds, the algorithm obtains some satisfying assignment of ϕ with probability, say, $> 1/3$.

It is easy to see that this algorithm fits into our framework. To be specific, we explain a bit more on the reduction $f_{\rm iso}$ and the set $L_{\rm iso}$ defined in [VV86]. For a given ϕ with n variables, $f_{\rm iso}$ uses a random seed s of length $r_{\rm iso}(n)$ for some polynomial $r_{\rm iso}$ to define *intuitively* some set C_s (which is independent from ϕ) and ask $L_{\rm iso}$ whether there exists a satisfying assignment for ϕ in C_s. More precisely, the value of $f_{\rm iso}(\phi)$ is simply (ϕ, s), and L is the set of pairs (ϕ, s) such that there exists some element in $C_s \cap W_\phi$, where $C_s \subseteq \{0,1\}^n$ is a certain polynomial-time recognizable set parameterized by s. Hence, D defined above is restated as

$$D \; = \; \{ \; (\phi, s, i, b) \mid \exists w \, [\, w \in C_s \cap W_\phi \text{ and } w_i = b \,] \; \}, \tag{7}$$

Now we can state the algorithm of Ben-David et al. in our framework follwing Definition 2 and Definition 3. We use $(\mathcal{Q}_{\rm BD}, \mathcal{F}_{\rm BD})$ to denite a witness finding algorithm implementing the algorithm of Ben-David et al. Fix any n. It asks $m := c_{\rm iso}n^2$ queries and uses a random seed of length $r := c_{\rm iso}n \cdot r_{\rm iso}(n)$. A random seed $s \in \{0,1\}^r$ is considered as a concatenation of $c_{\rm iso}n$ random seeds $s_1, ..., s_{c_{\rm iso}n}$ of length $r_{\rm iso}(n)$. For each k, i, b, such that $1 \le k \le c_{\rm iso}n$, $1 \le i \le n$, and $b \in \{0,1\}$, machine $\mathcal{Q}_{\rm BD}$ with a random seed s produces a query set $Q_{s,k,i,b}$ that is defined by

$$Q_{s,k,i,b} \;=\; \{\ v \mid v \in C_{s_k} \ \wedge\ v_i = b\ \},$$

and asks an oracle whether $Q_{s,k,i,b} \cap W \neq \emptyset$. (Precisely speaking, $\mathcal{Q}_{\mathrm{BD}}$ produces a circuit $C_{s,k,i,b}$ that accepts $Q_{s,k,i,b}$ and pass this circuit to the oracle as its (k,i,b)-th query.) In particular, when $W = W_\phi$ for a given $\phi \in 3\mathrm{SAT}_n$, then this query is the same as asking a query (ϕ, s_k, i, b) to the oracle set D. Then $\mathcal{F}_{\mathrm{BD}}$ is a machine that produces some $w \in W$ (if it exists) by using the answers from the oracle A_W to queries $Q_{s,k,1,0}, Q_{s,k,1,1}, ..., Q_{s,k,n,0}, Q_{s,k,n,1}$ for some k such that the isolation succeeds on W with the random seed s_k. (Note that the success of the isolation can be determined easily by looking at the answers on these queries.) It is easy to see that the success probability of this algorithm $(\mathcal{Q}_{\mathrm{BD}}, \mathcal{F}_{\mathrm{BD}})$ for randomly chosen s is $\Omega(1)$ for *any* nonemtpy witness set $W \subseteq \Omega$, including W_ϕ for any $\phi \in 3\mathrm{SAT}_n$.

Note that an oracle answering the above queries is in NP (if a witness set is W_ϕ for some $\phi \in 3\mathrm{SAT}$). Hence, this witness finding algorithm can be regarded as as an isolation algorithm asking queries to some NP oracle. While the isolation technique of Valiant-Vazirani is a many-one reduction, the above algorithm (or the algorithm of Ben-David et al.) is regarded as a truth-table reduction. Recently, Kabanets and Watanabe [KW11] discussed the limitation of the isolation technique of many-one reduction type in some blackbox framework. Our query lower bound result can be interpreted as the limitation of the truth-table reduction type isolation technique in a similar blackbox framework.

Example 2. (Decision vs. Search in Some Average-Case Scenario)
The motivation of the algorithm of Ben-David et al. is to show the relation between the average-case hardness of NP-type decision and search problems. With their algorithm, they showed roughly that if the NP set D defined by (7) is solvable in polynomial-time *on average*, then we can construct some polynomial-time algorithm that computes a satisfying assignment of formulas in 3SAT *on average*. More precisely, for the statements (a) and (b) defined under the following average-case scenario, they showed that (a) $\Rightarrow$ (b) holds[4].

(a) For any $d > 0$, some polynomial-time algorithm decides $(\phi, s, i, b) \in D$ correctly with probability $1 - n^{-d}$ when (ϕ, s, i, b) is taken uniformly random, ϕ from $3\mathrm{SAT}_n$, s from $\{0,1\}^{r_{\mathrm{iso}}(n)}$, i from $[n]$, and b from $\{0,1\}$;

(b) Some polynomial-time algorithm computes a satisfying assignment of ϕ with probability $1 - o(1)$ when ϕ is taken uniformly random from $3\mathrm{SAT}_n$.

Again we can explain this fact in our framework. First we show that the algorithm $(\mathcal{Q}_{\mathrm{BD}}, \mathcal{F}_{\mathrm{BD}})$ is $O(1/n^2)$-error tolerant, or more specifically $1/(10c_{\mathrm{iso}}n^2)$-error tolerant. Fix any n and any witness set $W \subseteq \Omega$, and consider any $1/(10c_{\mathrm{iso}}n^2)$-error bounded oracle $\widetilde{A}_W$. Here the set *Dom* of Definition 4 is

$$Dom \;=\; \{\ Q_{s,k,i,b} \mid s \in \{0,1\}^r, k \in [c_{\mathrm{iso}}n], i \in [n], b \in \{0,1\}\ \}.$$

Then by the standard averaging argument, we can show that for at least $4/5$ fraction of $s \in \{0,1\}^r$, $\widetilde{A}_W$ gives a correct answer to $Q_{s,k,i,b}$ for all $k \in [c_{\mathrm{iso}}n]$, $i \in [n]$, and $b \in \{0,1\}$. Recall

[4]The statement (b) is slightly weaker than the one stated in [BCGL92]; but it is easy to modify our argument to derive the original statement.

that there at least 1/3 fraction of $s \in \{0,1\}^r$ for which the algorithm succeeds to obtain some witness if all queries of type $Q_{s,k,i,b}$ are answered correctly (Example 1). Hence, the success probability of the algorithm when using $\widetilde{A}_W$ is at least $4/5 + 1/3 - 1 = \Omega(1)$. This proves the $1/(10c_{\rm iso}n^2)$-error tolerance of the algorithm.

Next we give a $1/(10c_{\rm iso}n^2)$-error tolerant oracle based on the assumption (a). Fix any n, and consider the set $3\mathrm{SAT}_n$ of all 3SAT formulas with n variables. Suppose that (a) holds; then we have some polynomial-time algorithm $\mathcal{A}$ that decides $(\phi, s, i, b) \in D$ correctly with probability $1 - n^{-3}$ when (ϕ, s, i, b) is given uniformly at random. Then again by the averaging argument, we can show that the proportion of "good" ϕ in $3\mathrm{SAT}_n$ satisfying

$$\Pr_{s:\{0,1\}^r,\, k:[c_{\rm iso}n],\, i:[n],\, b:\{0,1\}} [\ \mathcal{A} \text{ decides } (\phi, s_k, i, b) \in D \text{ correctly }\ ] \ \geq\ 1 - \frac{1}{10c_{\rm iso}n^2}$$

is $1 - o(1)$. Thus, for such a good ϕ, this $\mathcal{A}$ can be used as a $1/(10c_{\rm iso}n^2)$-error bounded oracle $\widetilde{A}_{W_\phi}$. Then by using the error tolerance property we just shown, the desired statement (b) is shown by the combined algorithm, i.e., $(\mathcal{Q}_{\rm BD}, \mathcal{F}_{\rm BD})$ using $\mathcal{A}$ as an oracle.

Example 3. (The Algorithm of Goldreich-Levin)
A one-way function is a polynomial-time computable function that is hard to invert. Consider any one-to-one[5] one-way function f. For simplicity, let us assume that f is a permutation on $\{0,1\}^n$ for each n; that is, f is a one-to-one function mapping every element in $\{0,1\}^n$ to some in $\{0,1\}^n$. For a given $y \in \{0,1\}^n$, computing the *preimage* $f^{-1}(y)$ of y, namely, w such that $f(w) = y$, is an NP-type search problem and w is regarded as the *unique* witness for $y \in \mathrm{Range}(f)$. Thus, we consider here the case where a witness set W is a singleton.

In order to show the "hard-core" property of some predicate, Goldreich and Levin [GL89] constructed an algorithm for computing $f^{-1}(y)$ by asking queries to some oracle set in NP. They also showed that the algorithm has a certain error tolerance property. We state their algorithm and their analysis in our framework. Below we sometimes regard a string in $\{0,1\}^n$ as a 0,1-vector with n coordinates, and for any $u, v \in \{0,1\}^n$, by $u + v$ and $u \odot v$, we mean their addition and inner product computed under modulo 2. For each $i \in [n]$, let e_i denote a string (or a vector) in $\{0,1\}^n$ that has 1 only at the i-th bit from the left (and the other bits are all 0).

First we recall the algorithm of Goldreich-Levin, see, e.g., [AB09, Gol01]. Consider any $n > 0$. For any $y \in \{0,1\}^n$, Goldreich-Levin algorithm computes $w = f^{-1}(y)$ by asking queries to the following NP set E.

$$E \ =\ \{\ (y, u) \mid \exists w\, [\, y = f(w) \wedge u \odot w = 1\,]\ \}.$$

For each $i \in [n]$, note that 0 ('no') or 1 ('yes') answer to the query "$(y, e_i) \in E$?" is nothing but w_i, the i-th bit of w. Thus, by asking these queries, we can get w. In order to be tolerant against errors, Goldreich-Levin algorithm uses some random set $U \subseteq \{0,1\}^n$ of m' elements (where m' is some odd number specified later) and asks queries "$(y, u + e_i) \in E$?" for all $u \in U$ and $i \in [n]$.

[5]The one-to-oneness is not necessary for the original result of Goldreich and Levin. But in our framework, even if a given function f is many-to-one, we need to use a singleton set for answering queries; that is, in our framework, the algorithm of Goldreich-Levin is for singleton sets. Thus, for simplifying the following explanation, we consider here only one-to-one functions.

Let $a(u,i)$ denote the 0,1-answer to the query "$(y, u+e_i) \in E$?" Suppose also that we have some list $\ell : U \to \{0,1\}^n$ of answers to the queries "$(y,u) \in E$?"; that is, $\ell(u)$ is 1 if $(y,u) \in E$ and 0 otherwise. Then we have $w_i = a(u,i) \oplus \ell(u)$. With an erroneous oracle, we may get some answer $\widetilde{\alpha}(u,i)$ for $a(u,i)$ with some errors. The algorithm computes w_i as the majority vote of $\widetilde{\alpha}(u,i) \oplus \ell(u)$ for all $u \in U$, thereby fixing such errors. But how do we get the list ℓ? Thanks to the specific way that the algorithm generates U, some m' candidates $\ell_1, ..., \ell_{m'}$ for this list are obtained, for which we can guarantee that the correct list always exists among them. Thus, the algorithm uses all ℓ_j's for ℓ, yielding m' candidates for w.

Now we follow Definition 2 to define a witness-list finding algorithm $(\mathcal{Q}_{\rm GL}, \mathcal{F}_{\rm GL})$ implementing the algorithm of Goldreich-Leving for inverting f. Fix any n and consider any singleton set $W_y = \{w | f(w) = y\}$; our problem here is to compute the unique witness $w \in W_y$. We use a random seed of length $r(n)$ for some polynomial r. For a given random seed $s \in \{0,1\}^r$, machine $\mathcal{Q}_{\rm GL}$ generates U of m' elements and asks the following query Q_u for each $u \in U$ and $i \in [n]$.

$$Q_{u,i} \ = \ \{ \ x \mid (u+e_i) \odot x = 1] \ \}.$$

Note that $A_{W_y}(Q_{u,i}) = 1 \Leftrightarrow Q_{u,i} \cap W_y \neq \emptyset \Leftrightarrow (y, u+e_i) \in E \Leftrightarrow a(u,i) = 1$. Thus, asking a query $Q_{u,i}$ in our framework is equivalent to asking a query $(y, u+e_i)$ to E in the original algorithm. We then define $\mathcal{F}_{\rm GL}$ as an algorithm that uses the lists $\{\ell_1, ..., \ell_{m'}\}$ explained above (that are polynomial-time computable from the seed s) to generate a list of candidates for $w \in W_y$ as explained above. That is, for each $k \in [m']$, $\mathcal{F}_{\rm GL}$ computes the k-th candidate $w^{(k)}$ as follows: for each $i \in [n]$, it first computes $b_u := \widetilde{A}_{W_y}(Q_{u,i}) \oplus \ell_k(u)$ for all $u \in U$ based on the answers from (possibly erroneous) oracle $\widetilde{A}_{W_y}$, and then it computes the i-th bit of $w^{(k)}$ as the majority value of $(b_u)_{u \in U}$.

The algorithm of Goldreich-Levin runs in polynomial-time and computes the inverse of f with probability $\Omega(1)$ by using any algorithm that solves E correctly with probability $1/2 + 1/n^{O(1)}$ under the uniform distribution. Here we explain this by showing a version of $(\mathcal{Q}_{\rm GL}, \mathcal{F}_{\rm GL})$ with $O(n^2/\varepsilon(n)^2)$ query complexity and $O(n/\varepsilon(n)^2)$ list size that is in fact $(1/2 - \varepsilon(n))$-error tolerant. (Then by the averaging argument similar to the one in Example 2, we can derive the desired success probability of the original algorithm for random $y \in \{0,1\}^n$.)

Let us fix n and any singleton witness set $W = \{w\}$. Precisely speaking, for each $\varepsilon > 0$, we consider the above $(\mathcal{Q}_{\rm GL}, \mathcal{F}_{\rm GL})$ defined with query complexity $m'n = c_{\rm GL} n^2/\varepsilon^2$ and list size $m' = c_{\rm GL} n/\varepsilon^2$ for some constant $c_{\rm GL}$, and we prove that it is $(1/2 - \varepsilon)$-error tolerant.

We first specify our error model. For any $i \in [n]$, consider the execution of the algorithm for computing (candidates for) w_i. Restricting on this computation, the set *Dom* is the family of query sets $Q_{u,i}$ that are asked for some $u \in U$ where U is some set generated by $\mathcal{Q}_{\rm GL}$. From the generation algorithm of U (which we omit in this paper), we know that U can be any subset of $\{0,1\}^n$ of m' elements; hence, any set $O_v := \{x | x \odot v = 1\}$ can be used as a query, and $Dom = \{O_v\}_{v \in \{0,1\}^n}$. Since the situation is the same for all $i \in [n]$, error probability is discussed by using the uniform distribution on this *Dom*.

For any singleton W, consider any $(1/2 + \varepsilon)$-error bounded oracle $\widetilde{A}_W$. That is, for randomly chosen $O_v \in Dom$, $\widetilde{A}_W(O_v)$ yields the correct answer, i.e., the decision for "$O_v \cap W \neq \emptyset$?" with probability $\geq 1/2 + \varepsilon$. Consider the execution of the algorithm for computing some i-th bit of w with this $\widetilde{A}_W$. For any $u \in \Omega$, let E_u denote the event that $\widetilde{A}_W(Q_{u,i}) \oplus \ell(u) = w_i$ holds, where $\ell(u) := \ell_k(u)$ with the correct ℓ_k in the list used by the algorithm. If u were chosen uniformly at random, then each query $Q_{u,i} = O_{u+e_i}$ would be independent and uniformly distributed in Dom, and hence, the event E_u holds with probability $\geq 1/2 + \varepsilon$ for each $u \in \Omega$ independently; thus, if m' elements of U were chosen independently, then the majority of $\widetilde{A}_W(Q_{u,i}) \oplus \ell(u)$ is w_i with high probability. Although U is generated "pseudo" randomly in the algorithm, due to its construction and the choice of $m' = c_{\mathrm{GL}} n/\varepsilon^2$, we can still guarantee that E_u holds for the majority of $u \in U$ with probability $\geq \alpha/n$ for some constant $\alpha > 0$. That is, with probability $\geq \alpha/n$, the majority of the values $\widetilde{A}_W(u,i) \oplus \ell_k(u)$ is w_i. Since this holds for any $i \in [n]$, by the averaging argument, it follows that the majority of the values $\widetilde{A}_W(u,i) \oplus \ell_k(u)$ is w_i for all $i \in [n]$ with probability $\geq \alpha$. This proves the algorithm running with $\widetilde{A}_W$ succeeds to get $w \in W$ with probability $\Omega(1)$ for any W.