On Multiple Input Problems in Property Testing

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
oded.goldreich@weizmann.ac.il

June 20, 2013

Abstract

We consider three types of multiple input problems in the context of property testing. Specifically, for a property $\Pi \subseteq \{0,1\}^n$, a proximity parameter ϵ, and an integer m, we consider the following problems:

1. Direct m-Sum Problem for Π and ϵ: Given a sequence of m inputs, output a sequence of m bits such that for each $i \in [m]$ the i^{th} bit satisfies the requirements from an ϵ-tester for Π regarding the i^{th} input; that is, for each i, the i^{th} output bit should be 1 (w.p. $\geq 2/3$) if the i^{th} input is in Π, and should be 0 (w.p. $\geq 2/3$) if the i^{th} input is ϵ-far from Π.
2. Direct m-Product Problem for Π and ϵ: Given a sequence of m inputs, output 1 (w.p. $\geq 2/3$) if all inputs are in Π, and output 0 (w.p. $\geq 2/3$) if at least one of the inputs is ϵ-far from Π.
3. The m-Concatenation Problem for Π and ϵ: Here one is required to ϵ-test the m-product of Π; that is, the property $\Pi^m = \{(x_1, ..., x_m) : \forall i \in [m]\ x_i \in \Pi\}$.

We show that the query complexity of the first two problems is $\Theta(m)$ times the query complexity of ϵ-testing Π, whereas (except in pathological cases) the query complexity of the third problem is almost of the same order of magnitude as the query complexity of the problem of ϵ-testing Π. All upper bounds are shown via efficient reductions.

We also consider the nonadaptive and one-sided error versions of these problems. The only significant deviation from the picture in the general (adaptive and two-sided error) model is that the *one-sided error* query complexity of the Direct Product Problem equals $\Theta(m)$ times the (two-sided error) query complexity of ϵ-testing Π plus $\Theta(1)$ times the *one-sided error* query complexity of ϵ-testing Π.

Keywords: Property Testing, Direct Sum Theorems, Direct Product Theorems, Composition Theorems, Adaptive vs Nonadaptive queries, One-Sided Error vs Two-Sided Error, Yao's MiniMax Principle, Levin's Economical Work Investment Strategy.

Contents

1 Introduction

In the last couple of decades, the area of property testing has attracted much attention (see, e.g., a couple of recent surveys [11, 12] and a collection of texts [4]). Loosely speaking, property testing typically refers to sub-linear time probabilistic algorithms for deciding whether a given object has a predetermined property or is far from any object having this property. Such algorithms, called testers, obtain local views of the object by performing queries; that is, the object is seen as a function and the testers get oracle access to this function (and thus may be expected to work in time that is sub-linear in the length of the object).

Indeed, it will be useful to bear in mind that property testing problems are promise problems. The promise is that the object either has the property or is far from having the property, and the task is to distinguish these two cases.

A basic question that was raised in several complexity theoretic contents is that of the complexity of solving several instances of a problem as compared to the complexity of solving a single instance of that problem. Typically, this question appears in two flavors: The first type requires to provide answers to all instances (typically called a direct sum problem), whereas a second type requires to provide a (possibly Boolean) function of these answers (typically called a direct product problem). For example, in the context of Cryptography, the amplification of one-way functions (cf. [2, Sec.2.3]) belongs to the first type, whereas Yao's XOR Lemma (cf. [7]) or the "Selective XOR Lemma" of [6] (cf. [2, Sec. 2.5.2] or [3, Sec. 7.2.1.2]) belongs to the second type.

In this paper, we consider the aforementioned type of questions in the context of property testing. As we shall see, in this context, two natural versions of the direct product problem arise. Hence, we shall consider three types of multiple input problems in the context of property testing, and cast each of them as a promise problem.

1.1 The three problems

In all cases we refer to a basic property $\Pi \subseteq \{0,1\}^n$ and to a proximity parameter, denoted ϵ, which together determine the promise problem; that is, the basic problem, called ϵ-testing Π, is distinguishing between inputs in Π and inputs in $\overline{\Gamma}_\epsilon(\Pi)$, where $\overline{\Gamma}_\epsilon(\Pi)$ denotes the set of n-bit strings that are ϵ-far from Π (i.e., $x \in \overline{\Gamma}_\epsilon(\Pi)$ iff every $y \in \Pi$ differs from x on at least ϵn bits). (As usual, the ϵ-tester is required to be correct, on each input in $\Pi \cup \overline{\Gamma}_\epsilon(\Pi)$, with probability at least 2/3.) In all cases, we refer to a sequence of m inputs for the ϵ-testing problem.

A Direct Sum Problem. Here the promise is that each of the m inputs is in $\Pi \cup \overline{\Gamma}_\epsilon(\Pi)$, and we are required to solve the basic problem for each of the m inputs. That is, on input $(x_1, ..., x_m) \in \{0,1\}^{mn}$, we are required to output $(\sigma_1, ..., \sigma_m) \in \{0,1\}^m$ such that for every $i \in [m]$ the following holds:

1. If $x_i \in \Pi$, then $\Pr[\sigma_i = 1] \geq 2/3$.
2. If $x_i \in \overline{\Gamma}_\epsilon(\Pi)$, then $\Pr[\sigma_i = 0] \geq 2/3$.

In other words, we are required to solve m instances of the promise problem, and each instance should be solved with (proximity and error) parameters as in the basic problem.

We show that *the query complexity of the Direct Sum Problem is* $\Theta(m)$ *times the query complexity of* ϵ*-testing* Π. The upper bound is obvious, and our focus is on the lower bound.

A Direct Product Problem. Again, the promise is that each of the m inputs is in $\Pi \cup \overline{\Gamma}_\epsilon(\Pi)$, but here we are required to output the conjunction of the m answers referred to in the Direct Sum Problem. That is, on input $(x_1, ..., x_m)$, we should output a bit σ such that the following holds:

1. If $x_i \in \Pi$ holds for each $i \in [m]$, then $\Pr[\sigma = 1] \geq 2/3$.
2. If $x_i \in \overline{\Gamma}_\epsilon(\Pi)$ holds for some $i \in [m]$, then $\Pr[\sigma = 0] \geq 2/3$.

In other words, we are required to distinguish Π^m (i.e., the m-way Cartesian product of Π) from $(\Pi \cup \overline{\Gamma}_\epsilon(\Pi))^m \setminus \Pi^m$. Indeed, this version of a direct product problem is more natural in the current contents than an alternative version that refers to the exclusive or of the aforementioned m answers. In particular, the current direct product problem is related to the concatenation problem that we consider next.

We show that *the query complexity of the Direct Product Problem is* $\Theta(m)$ *times the query complexity of* ϵ*-testing* Π. Note that an upper bound with an $O(m \log m)$ factor is obvious (using error reduction), and something more seems needed in order to to get an $O(m)$ factor.

Indeed, neither the upper bound nor the lower bound is directly related to the corresponding bound for the Direct Sum Problem. In the case of the upper bound, the point is that in the Direct Sum Problem we are only guaranteed that each of the m answers is correct with probability at least $2/3$, whereas the straightforward solution to the Direct Product Problem requires all answers to be correct, which yields an extra $O(\log m)$ factor (for error reduction). The same issue arises when trying to derive the lower bound for the Direct Sum Problem from the one for the Direct Product Problem: Starting from an $\Omega(m)$-factor lower bound for the Direct Product Problem, one only derives an $\Omega(m/\log m)$ factor for the Direct Sum Problem. As hinted above, in both cases, the extra $O(\log m)$ factor can be eliminated (see Lemma 1.1). We shall use this fact to upper bound the complexity of the Direct Product Problem, but avoid using it for deriving a lower bound for the Direct Sum Problem (because the proof of the lower bound for the Direct Product Problem extends the proof of the lower bound for the Direct Sum Problem).

A Concatenation Problem. Here, the promise is that the sequence of m inputs (or the concatenation of the m inputs) is in $\Pi^m \cup \overline{\Gamma}_\epsilon(\Pi^m)$, where $\overline{\Gamma}_\epsilon(\Pi^m)$ denotes the set of mn-bit strings that are ϵ-far from Π^m, and we are required to distinguish the two cases. That is, on input $(x_1, ..., x_m)$, we should output a bit σ such that the following holds:

1. If $x_i \in \Pi$ holds for each $i \in [m]$, then $\Pr[\sigma = 1] \geq 2/3$.
2. If there exists $\epsilon_1, ..., \epsilon_m$ that sum-up to $m\epsilon$ such that $x_i \in \overline{\Gamma}_{\epsilon_i}(\Pi)$ holds for each $i \in [m]$, then $\Pr[\sigma = 0] \geq 2/3$.

In other words, we are required to ϵ-test the property Π^m (i.e., the m-way Cartesian product of Π).

We show that *the query complexity of the Concatenation Problem is the almost same as the query complexity of* ϵ*-testing* Π*, provided that the latter increases at least linearly with* $1/\epsilon$, where "almost the same" allows a polylogarithmic slackness factor. Furthermore, if for some $c > 1$ the query complexity of ϵ-testing Π increases at least linearly with $1/\epsilon^c$, then up to a constant factor the query complexity of the Concatenation Problem is the same as the query complexity of ϵ-testing Π. We comment that in all reasonable cases the query complexity of ϵ-testing increases at least linearly with $1/\epsilon$, and an increase rate of at least $1/\epsilon^2$ is very common.

1.2 Nonadaptive queries and one-sided error versions

We also consider the nonadaptive and one-sided error versions of the problems discussed in Section 1.1. The results that we obtain differ from the those obtained in the general model (of adaptive and two-sided error algorithms) in two cases.

Most importantly, it turns out that the *one-sided error* query complexity of the Direct Product Problem equals $\Theta(m)$ times the *two-sided error* query complexity of ϵ-testing Π plus $\Theta(1)$ times the *one-sided error* query complexity of ϵ-testing Π. The point is that the two-sided error query complexity of ϵ-testing Π may be significantly lower than its one-sided error query complexity.[1]

The second case refers to the *nonadaptive* query complexity of the Direct Product Problem, where we leave a small gap: We show that the nonadaptive query complexity of the Direct Product Problem is at least $\Omega(m)$ times the nonadaptive query complexity of ϵ-testing Π, and at most $O(m \log m)$ the latter amount.

1.3 Two comments

Computational complexity. Our exposition focuses on the query complexity of problems, which is natural since our main focus is on lower bounds. We stress that all our upper bounds are obtained by computationally efficient reductions, yielding computationally efficient algorithms for the multi-instance problems whenever such algorithms are given for the basic testing problem.

A somewhat tedious comment: The Direct Sum and Direct Product problems were stated in Section 1.1 as promise problems regarding a sequence in $(\Pi \cup \overline{\Gamma}_\epsilon(\Pi))^m$. An alternative statement, used in the technical sections, refers to all m-ary sequences and adapts the output requirements accordingly. Specifically, in Section 3 the Direct Sum Problem is defined for all m-ary sequences but requirements are made with respect to the i^{th} answer only when the i^{th} instance is in $\Pi \cup \overline{\Gamma}_\epsilon(\Pi)$. In Section 4 the Direct Product Problem is defined for all m-ary sequences but requirements are made only with respect to sequences that are either in Π^m or contain some instance in $\overline{\Gamma}_\epsilon(\Pi)$. In all cases, our lower bounds hold for the more restricted versions (as stated in Section 1.1) whereas our upper bounds hold also for the relaxed versions (as stated in the technical sections).

1.4 Techniques

Our lower bounds (for the query complexity of the Direct Sum and Product problems) capitalize on the fact that in the context of (computationally unbounded) oracle machines it is easy to decouple the computation on multiple inputs to a sequence of (possibly related) computations on single inputs. In particular, the queries to the different inputs are easily (if not trivially) distinguishable. So the only issue at hand is the allocation of resources (i.e., queries) among the multiple computations; that is, the algorithm solving the multiple-instance problem may allocate its resources in an unequal manner.[2]

In the case that the algorithm solving the multiple-instance problem is nonadaptive (i.e., its queries are determined by its randomness, obliviously of the answers to "prior" queries), we may proceed as follows: First, we identify an index $i \in [m]$ such that the expected number of queries made to the i^{th} instance is at most a $1/m$ fraction of the total number of queries made, denoted

[1] Consider, for example, the set Π of n-bit strings having at least $n/2$ one-entries. Other examples include ρ-clique in the dense graphs model [5], and cycle-freeness in the bounded-degree graph model [8].

[2] Indeed, if the algorithm solving the multiple-instance problem always allocates equal resources to the different instances, then the lower bound follows easily.

q. Next, we truncate executions that make more than $10 \cdot q/m$ queries, obtaining an ϵ-tester for Π that errs with probability at most $(1/3) + 0.1 < 0.45$. Finally, we apply error reduction, and conclude that $O(q/m)$ is lower-bounded by the query complexity of ϵ-testing Π.

However, in general, the algorithm solving the multiple-instance problem may be adaptive (see the proofs of Lemma 1.1 and Theorem 6.2 for demonstrations of the possible benefit of adaptivity in the context of multiple-instance problems). In this case the distribution of the algorithm's queries among the m instances may depend on answers to prior queries. We may want to resolve this problem by looking at a "typical" (or "random") sequence of m instances, but the question is what distribution of instances should we consider. At this point, the MiniMax Principle (cf. Yao [15] following von Neumann [14]) comes handy.

Specifically, our lower bounds for the Direct Sum and Product problems relies on the fact that if the query complexity of ϵ-testing Π is (at least) q, then there exists a distribution of n-bit strings, denoted D, such that every oracle machine M that makes less than q queries errs with probability greater than $1/3$ on the distribution D; that is,

$$\Pr_{x \leftarrow D}\left[(x \in \Pi \wedge M^x = 1) \vee (x \in \overline{\Gamma}_\epsilon(\Pi) \wedge M^x = 0)\right] < 2/3, \tag{1}$$

where $x \leftarrow D$ means that x is selected at random according to D. Indeed, this is the actual contents of Yao's (or von Neumann's) MiniMax Principle.[3]

The lower bound on the Direct Sum Problem is obtained by considering the m-way product of the distribution D. The lower bound on the Direct Product Problem is obtained by considering the m-way product of the distribution D conditioned on having at most one instance in $\overline{\Gamma}_\epsilon(\Pi)$.

The upper bound on the Concatenation Problem uses the fact that the distance in this problem is the average of the distances on the m basic problems. Thus, the tester consists of sampling a few of these basic problems and testing them while using suitable values of the proximity parameter. The economical way in which this is done is inspired by a private communication with Leonid Levin (in mid-1980s).[4]

Regarding the connection between the Direct Sum and Direct Product problems, as noted above, the m-way Direct Product Problem can be easily reduced to the m-way Direct Sum Problem using $O(\log m)$ calls (which are employed for error reduction). Our improved upper bound for the Direct Product Problem is obtained by the following general result, which is implicit in the work of Feige *et al.* [1].[5]

Lemma 1.1 (reducing direct product to direct sum, following [1, Thm. 2.7]): *The m-way direct product problem is reducible to $O(j)$ instances of the $2^{-(j-1)} \cdot m$-way direct sum problem, for every $j = 1, ..., \lceil \log_2 m \rceil$.*

Hence, the m-way direct product problem can be solved at the cost of $\sum_{j=1}^{\log_2 m} O(j \cdot 2^{-j} m) = O(m)$ instances of the basic problem. We note that this reduction is quite generic: It holds not only when

[3] Let us mention, in passing, that we have always objected to the practice of attributing the converse of the above to Yao [15] (or to von Neumann [14]). That is, the fact that the existence of D such that Eq. (1) holds implies that the query complexity of ϵ-testing Π is at least q is a triviality. What is non-trivial is the fact that this method of obtaining lower bounds is actually "complete" (i.e., it yields the best possible lower bounds). This non-trivial direction is the one we use here. For further discussion, see Appendix A.1.

[4] The idea appeared in [9, Sec. 9], and we do not recall a prior use of it. Following [9], this idea was also used in [6, Lem. 3] (see also [2, Clm. 2.5.4.1]). Within the context of property testing, this idea was first used in [8] (see Lemma 3.3 in the proceeding version and Lemma 3.6 in the journal version). For further discussion see Appendix A.2.

[5] The result of Feige *et al.* [1] is stated in terms of computing an m-wise AND by a noisy decision tree. A simpler procedure is provided by Newman [10, Obs. 2.2]. Our procedure is different from both.

the basic problem is ϵ-testing some property, but rather with respect to any randomized procedures for solving decision problems with constant error probability.

Proof: The issue is that when we invoke the direct sum algorithm on a sequence of m instances for the basic problem Π, we may get many 0-answers even if all instances are in Π (and so we cannot distinguish this case from the case in which the sequence of instances contains few instances in $\overline{\Gamma}_\epsilon(\Pi)$). Our solution is to declare the instances for which a 0-answer was obtained as candidates for being in $\overline{\Gamma}_\epsilon(\Pi)$, and apply the direct sum algorithm only to the surviving candidates. This process is iterated, as long as the set of candidates is not too big, which may happen only if this set contains many instances in $\overline{\Gamma}_\epsilon(\Pi)$. For this process to work, we reduce the error probability in the various iterations such that in the j^{th} iteration we use $O(j)$ repeatitions and have error probability 2^{-j} (or so) per each instance. This implies that the set of false candidates (i.e., candidates that are actually in Π) does shrink in each iteration, while each instance in $\overline{\Gamma}_\epsilon(\Pi)$ survives all iterations with high probability (e.g., with probability at least 0.9). Details follow.

Given an instance $(x_1, ..., x_m)$ for the direct product problem, we proceed in $\ell = \lceil \log_2(3m) \rceil$ iterations, while maintaining a set $I \subseteq [m]$ of candidates (for being in $\overline{\Gamma}_\epsilon(\Pi)$). Initially, $I = [m]$. In the j^{th} iteration, if $|I| > 2^{-(j-1)} \cdot m$, then we output 0. Otherwise, we invoke the $|I|$-way direct product algorithm on $(x_{i_1}, ..., x_{i_t})$, where $I = \{i_1, ..., i_t\}$, for $O(j)$ times, and rule by majority on each of the x_i's (with $i \in I$). We keep i in I if the majority vote on x_i is 0. If, at the end of any iteration, the set I becomes empty, then we output 1.

Specifically, in the j^{th} iteration, the direct product algorithm is invoked for $O(j)$ times in order to guarantee that the majority vote (on each x_i) is correct with probability at least $1 - 2^{-(j+3)}$. Hence, each $x_i \in \overline{\Gamma}_\epsilon(\Pi)$ has a fair chance to survive all iterations, in which case I will become too big at some iteration (since $2^{-(\ell-1)} < 1/m$). On the other hand, with very high probability, the vote on almost all inputs is correct. Hence, if all x_i's are in Π, then we expect the set I to be cut by a factor of at least two in each iteration, and so eventually $I = \emptyset$. Further details follow.

We first note that if any of the x_i's is in $\overline{\Gamma}_\epsilon(\Pi)$, then i remains in I throughout all iterations, with probability at least $1 - \sum_{j=1}^{\ell} 2^{-(j+3)} > 7/8$. In this case the algorithm outputs 0, because, for some $j \geq 2$ (possibly for $j = \ell$), at the beginning of the j^{th} iteration it holds that $|I| > 2^{-(j-1)} \cdot m$.

On the other hand, if $(x_1, ..., x_m) \in \Pi^m$, then with probability at least $1 - \sum_{j=1}^{\ell} 2^{-(j+2)} > 3/4$ in each iteration j it holds that $|I| \leq 2^{-(j-1)} \cdot m$. (This is the case since the expected size of I is cut by a factor of $2^{-(j+3)}$, and so with probability at least $1 - 2^{-(j+2)}$ it is cut by half.) In this case the algorithm outputs 1, because for some $j \geq 2$ (possibly for $j = \ell$) at the end of the j^{th} iteration it holds that I is empty. ■

1.5 Organization

Following a short preliminaries section, we proceed to the study of the three problems described above: The Direct Sum Problem is studied in Section 3, the Direct Product Problem is studied in Section 4, and the Concatenation Problem is studied in Section 5. In Section 6 we consider nonadaptive and one-sided error versions of these problems.

In the appendix, we elaborate on two comments that were made in Section 1.4 (see Footnotes 3 and 4): Section A.1 discusses Yao's MiniMax Principle, whereas Section A.2 surveys a general method (which we call Levin's Economical Work Investment Strategy) that underlies some of the saving obtained in Section 5.

2 Preliminaries

For sake of simplicity, we present all results in terms of properties of fixed-length strings (i.e., n is fixed), and while referring to testing them with respect to a fixed value of the proximity parameter, denoted ϵ. Nevertheless, both parameters should be viewed as generic (and thus varying).

Definition 2.1 (property testing): *Let $\Pi \subseteq \{0,1\}^n$ and $\epsilon > 0$. An ϵ-tester for Π is a randomized oracle machine T that satisfies the following two conditions.*

1. *If $x \in \Pi$, then $\Pr[T^x = 1] \geq 2/3$.*
2. *If $x \in \{0,1\}^n$ is ϵ-far from Π, then $\Pr[T^x = 0] \geq 2/3$, where the distance between strings is the fraction of bits on which they disagree and the distance to a set is the distance to the closest element in the set. That is, x is ϵ-far from Π if and only if every $y \in \Pi$ differs from x on at least ϵn bits.*

The query complexity of T is the maximum number of queries that T makes, when the maximization is over all $x \in \{0,1\}^n$ and over the coin tosses of T. The query complexity of ϵ-testing Π, denoted $\mathtt{Q}_\epsilon(\Pi)$, is the minimum query complexity of all ϵ-testers for Π.

Indeed, ϵ-testing is the promise problem that consists of distinguishing inputs in Π from inputs that are ϵ-far from Π.

3 The Direct Sum Problem

Recall that in this problem, on input $(x_1, ..., x_m) \in \{0,1\}^{mn}$, we should output $(\sigma_1, ..., \sigma_m) \in \{0,1\}^m$ such that for every $i \in [m]$ the following holds:

1. If $x_i \in \Pi$, then $\Pr[\sigma_i = 1] \geq 2/3$.
2. If $x_i \in \overline{\Gamma}_\epsilon(\Pi)$, then $\Pr[\sigma_i = 0] \geq 2/3$.

Let us denote this problem by $\mathtt{DS}^m_\epsilon(\Pi)$.

Theorem 3.1 (The Direct Sum Theorem): *For every property Π, proximity parameter ϵ, and integer m, the query complexity of $\mathtt{DS}^m_\epsilon(\Pi)$ is $\Theta(m \cdot \mathtt{Q}_\epsilon(\Pi))$.*

Here and in all our results the hidden constants in the Θ notation are universal (i.e., are independent of Π, ϵ and m). Ditto for the O and Ω notations.

Proof: The upper bound holds by merely invoking the ϵ-tester, T, of Π for m times; that is, on input $(x_1, ..., x_m)$, we output $(T^{x_1}, ..., T^{x_m})$.

Turning to the lower bound, we start by invoking the MiniMax Principle of von Neumann [14] as adapted by Yao [15]. That is, let $q \stackrel{\text{def}}{=} \mathtt{Q}_\epsilon(\Pi)$, and consider a two-player zero-sum game between an algorithmic player and an adversarial player. In the game, the algorithmic player selects (randomly or deterministically) a deterministic oracle machine M that makes at most $q - 1$ queries and the adversarial player selects (randomly or deterministically) an input x. The algorithmic player wins if and only if either $x \notin (\Pi \cup \overline{\Gamma}_\epsilon(\Pi))$ or M^x outputs the correct answer.

By our hypothesis, the algorithmic player has no strategy that guarantees winning with probability at least $2/3$; that is, for every distribution on deterministic oracle machines $\mathcal{M}$ that make

at most $q-1$ queries, there exists a string $x \in \Pi \cup \overline{\Gamma}_\epsilon(\Pi)$ such that when selecting $M \leftarrow \mathcal{M}$ the probability that M^x is correct is smaller than 2/3. The MiniMax Principle asserts that, in this case, there exists a distribution of inputs in $\Pi \cup \overline{\Gamma}_\epsilon(\Pi)$ on which each (deterministic) oracle machine M that make at most $q-1$ queries fails with probability greater than 1/3. Let us denote this distribution by D.

Now, consider an arbitrary (randomized) oracle machine $\overline{M}_0$ of query complexity q_0 that solves $\mathtt{DS}^m_\epsilon(\Pi)$. By straightforward amplification, we obtain a machine $\overline{M}$ of query complexity $q_1 = 10q_0$ having an error probability of at most 1/6 on each answer. That is, letting $\overline{M}^{\overline{x}}_i$ denote the i^{th} bit in $\overline{M}^{\overline{x}}$, for each $\overline{x} = (x_1, ..., x_m)$ and each $i \in [m]$, the following holds:

1. If $x_i \in \Pi$, then $\Pr[\overline{M}^{\overline{x}}_i = 1] \geq 5/6$.
2. If $x_i \in \overline{\Gamma}_\epsilon(\Pi)$, then $\Pr[\overline{M}^{\overline{x}}_i = 0] \geq 5/6$.

Now, consider the m-way Cartesian product of D, denoted D^m, and consider the execution of $\overline{M}$ on an input drawn from D^m. Let $i \in [m]$ be such that the expected number of queries that $\overline{M}$ makes to its i^{th} input is at most $q' = q_1/m$, where the expectation is taken over D^m as well as over the coins of $\overline{M}$. Then, with probability at least 5/6, machine $\overline{M}$ makes at most $6q'$ queries to its i^{th} input. It follows that, with probability at least $(5/6) - (1/6) = 2/3$, machine $\overline{M}$ makes at most $6q'$ queries to its i^{th} input and yet answers correctly regarding this input.

Using $\overline{M}$, we obtain a randomized oracle machine M' that makes at most $6q'$ queries and solves the basic problem on D with probability at least 2/3, where the probability is taken over both D and the internal coins of M': Machine M' just emulates the execution of $\overline{M}$, while using its own input as the i^{th} input of $\overline{M}$, emulating the other inputs (by generating $m-1$ samples according to D), and terminating the execution of $\overline{M}$ if $\overline{M}$ tries to make more than $6q'$ queries to its i^{th} input (which happens with probability at most 1/6). Using an averaging argument, we obtain a deterministic oracle machine M'' that makes at most $6q'$ queries and succeeds on D with probability at least 2/3. It follows that $6q' > q-1$ (or $6q' \geq q$), and thus $q_0 = \frac{q_1}{10} = \frac{m \cdot q'}{10} \geq \frac{m \cdot q}{60}$. ■

4 The Direct Product Problem

Recall that in this problem, on input $(x_1, ..., x_m) \in \{0,1\}^{mn}$, we should output $\sigma \in \{0,1\}$ such that the following holds:

1. If for every $i \in [m]$ it holds that $x_i \in \Pi$, then $\Pr[\sigma = 1] \geq 2/3$.
2. If there exists $i \in [m]$ such that $x_i \in \overline{\Gamma}_\epsilon(\Pi)$, then $\Pr[\sigma = 0] \geq 2/3$.

Let us denote this problem by $\mathtt{DP}^m_\epsilon(\Pi)$.

Theorem 4.1 (The Direct Product Theorem): *For every property Π, proximity parameter ϵ, and integer m, the query complexity of $\mathtt{DP}^m_\epsilon(\Pi)$ is $\Theta(m \cdot \mathtt{Q}_\epsilon(\Pi))$.*

Proof: The upper bound follows by combining Lemma 1.1 with the upper bound of Theorem 3.1: That is, we use the reduction of the lemma, and apply the straightforward algorithm asserted by the theorem. Turning to the lower bound, we start as in the proof of Theorem 3.1, and let D denote the corresponding "hard" distribution. Actually, we consider ϵ-testers for Π that are (only) correct with probability at least 0.51, and derive a lower bound of $\Omega(\mathtt{Q}_\epsilon(\Pi))$ on their query complexity, denoted q, because otherwise a contradiction follows by a straightforward amplification.

Let D' denote the distribution obtained from D by conditioning that the string is in Π, and likewise D'' is obtained by conditioning that the string is in $\overline{\Gamma}_\epsilon(\Pi)$. For each $i \in [m]$, we will consider the distribution $D^{(i)}$ that consists of the Cartesian product of $m-1$ copies of D' and a single copy of D'' placed in the i^{th} position. We shall also consider the m-way Cartesian product of D', denoted $(D')^m$.

Given any oracle machine $\overline{M}_0$ of query complexity q_0 that solves $\mathtt{DP}^m_\epsilon(\Pi)$, we consider its amplification to a machine $\overline{M}$ of query complexity $q_1 = O(q_0)$ having error probability at most 0.01 on each input. We first consider the invocation of $\overline{M}$ on inputs drawn from $(D')^m$, and let $i \in [m]$ be such that the expected number of queries that $\overline{M}$ makes to its i^{th} input is at most $q' = q_1/m$. Hence, with probability at least 0.99, machine $\overline{M}$ makes at most $100q'$ queries to the i^{th} input. We now consider two cases regarding the execution of $\overline{M}$ on an input drawn from $D^{(i)}$:

1. If when invoked on $D^{(i)}$, with probability at least 5/6, machine $\overline{M}$ makes at most $100q'$ queries to the i^{th} input, then we proceed as in the proof of Theorem 3.1. Specifically, in this case the $100q'$-query truncated executions of $\overline{M}$ yield a machine M'' for ϵ-testing Π, and so $100q' > q-1$ must hold (which in turn implies $q_0 = \Omega(m \cdot q)$).

2. Otherwise (with probability at least 1/6, machine $\overline{M}$ makes more than $100q'$ queries to the i^{th} input), we can use this as an indication to whether $\overline{M}$ runs on $(D')^m$ or on $D^{(i)}$, which in turn yields an ϵ-tester for Π. Specifically, consider an alternative randomized machine T that on input $x \in \{0,1\}^n$ invokes $\overline{M}$ on input $(x_1, ..., x_{i-1}, x, x_{i+1}, ..., x_m)$, where $(x_1, ..., x_m) \leftarrow (D')^m$, outputs 0 if $\overline{M}$ tries to make more than $100q'$ queries to x, and otherwise outputs the outcome of a coin with bias 0.55 towards 1. Then, $\Pr_{x \leftarrow D'}[T^x = 1] \geq 0.99 \cdot 0.55 > 0.51$, whereas $\Pr_{x \leftarrow D''}[T^x = 0] \geq \frac{5}{6} \cdot 0.45 + \frac{1}{6} > 0.51$. Thus, the probability that T decides D correctly exceeds 0.51, and we again derive $100q' > q-1$.

The theorem follows. ■

5 The Concatenation Problem

Recall that in this problem, on input $(x_1, ..., x_m) \in \{0,1\}^{mn}$, we should output $\sigma \in \{0,1\}$ such that the following holds:

1. If for every $i \in [m]$ it holds that $x_i \in \Pi$ (equiv., $(x_1, ..., x_m) \in \Pi^m$), then $\Pr[\sigma = 1] \geq 2/3$.

2. If there exists $\epsilon_1, ..., \epsilon_m$ that sum-up to $m\epsilon$ such that for every $i \in [m]$ it holds that $x_i \in \overline{\Gamma}_{\epsilon_i}(\Pi)$ (equiv., $(x_1, ..., x_m) \in \overline{\Gamma}_\epsilon(\Pi^m)$), then $\Pr[\sigma = 0] \geq 2/3$.

Let us denote this problem by $\mathtt{CP}^m_\epsilon(\Pi)$.

Theorem 5.1 (The First Concatenation Theorem): *Suppose that $\mathtt{Q}_\epsilon(\Pi)$ increases at least linearly with $1/\epsilon$; that is, $\mathtt{Q}_{\epsilon/2}(\Pi) \geq 2 \cdot \mathtt{Q}_\epsilon(\Pi)$ for every $\epsilon > 0$. Then, the query complexity of $\mathtt{CP}^m_\epsilon(\Pi)$ is $O(\log(1/\epsilon))^3 \cdot \mathtt{Q}_{\epsilon/4}(\Pi) = \widetilde{O}(\mathtt{Q}_{\epsilon/4}(\Pi))$.*

Proof: Fixing any $(x_1, ..., x_m)$, let $\delta_i \in [0,1]$ denote the distance of x_i from Π. The key observation is that if $\sum_{i \in [m]} \delta_i \geq m \cdot \epsilon$, then for $\ell = \lceil \log_2(1/\epsilon) \rceil + 1$ there exists some $j \in [\ell]$ such that $|\{i \in [m] : \delta_i \in [2^{-j}, 2^{-(j-1)}]\}| \geq m \cdot 2^j \epsilon / 4\ell$ (see Fact A.1).[6] But in such a case, sampling $O(\ell/2^j\epsilon)$

[6] Consider the uniform distribution over $[m]$, and let $q(s) = \delta_s$.

indices i, and 2^{-j}-testing each input x_i w.r.t Π will do (i.e., with high constant probability, we will sample an instance that is 2^{-j}-far from Π). Details follow.

First, let us spell out the proposed ϵ-tester (for Π^m). On input $(x_1, ..., x_m)$, for every $j \in [\ell]$, the tester samples $O(\ell/2^j\epsilon)$ indices i, and 2^{-j}-tests each input x_i w.r.t Π, with error probability ϵ^2 (rather than 1/3).[7] Hence, we make $O(\ell) \cdot \mathtt{Q}_{2^{-j}}(\Pi)$ queries to each x_i that is sampled in the j^{th} iteration, and can neglect the probability that we obtained a wrong result for any x_i in any iteration. The tester accepts if and only if all the aforementioned tests answer 1 (i.e., all sampled x_i's were verified as being in Π).

Turning to the analysis of the above tester, we first note that the number of queries made by it is upper-bounded by $q \stackrel{\text{def}}{=} \sum_{j\in[\ell]} O(\ell/2^j\epsilon) \cdot O(\ell \cdot \mathtt{Q}_{2^{-j}}(\Pi))$, whereas $\mathtt{Q}_{2^{-j}}(\Pi) \leq \frac{\epsilon/4}{2^{-j}} \cdot \mathtt{Q}_{\epsilon/4}(\Pi)$ (by the theorem's hypothesis). Thus, $q \leq \sum_{j\in[\ell]} O(\ell^2) \cdot \mathtt{Q}_{\epsilon/4}(\Pi)$, which meets the asserted complexity bound. Next, note that if $(x_1, ..., x_m) \in \Pi^m$, then each of the tests answers 1 with probability at least $1 - \epsilon^2$, whereas the number of tests is $\sum_{j\in[\ell]} O(\ell/2^j\epsilon) = O(\ell/\epsilon)$. On the other hand, if $(x_1, ..., x_m) \in \overline{\Gamma}_\epsilon(\Pi^m)$, then, for some $j \in [\ell]$, with probability at least 3/4, some $x_i \in \overline{\Gamma}_{2^{-j}}(\Pi)$ is sampled and 2^{-j}-tested (and so answered 0 with probability at least $1 - \epsilon^2 > 0.99$). The theorem follows. ■

Theorem 5.2 (The Second Concatenation Theorem): *Suppose that, for some constant $c > 1$, it holds that $\mathtt{Q}_\epsilon(\Pi)$ increases at least linearly with $1/\epsilon^c$; that is, $\mathtt{Q}_{\epsilon/2}(\Pi) \geq 2^c \cdot \mathtt{Q}_\epsilon(\Pi)$ for every $\epsilon > 0$. Then, the query complexity of $\mathtt{CP}^m_\epsilon(\Pi)$ is $O(\mathtt{Q}_{\epsilon/4}(\Pi))$.*

Proof: We follow the proof of Theorem 5.1, while using a slightly different analysis of the various "buckets" (or the sets) $B_j \stackrel{\text{def}}{=} \{i \in [m] : \delta_i \in [2^{-j}, 2^{-(j-1)}]\}$. Specifically, for $\ell = \lceil \log_2(1/\epsilon) \rceil + 1$ and $p(j) \stackrel{\text{def}}{=} (\ell + 5 - j)^{-2}$, we first prove that *if $\sum_{i\in[m]} \delta_i \geq m \cdot \epsilon$, then there exists some $j \in [\ell]$ such that $|B_j| \geq m \cdot p(j) \cdot 2^j\epsilon$*. This is essentially proved in Fact A.2, and the argument is adapted next (for the reader's convenience). Indeed, assuming towards the contradiction that for every $j \in [\ell]$ it holds that $|B_j| < m \cdot p(j) \cdot 2^j\epsilon$, we get

$$\begin{aligned}
\sum_{i\in[m]} \delta_i &< \left(\sum_{j\in[\ell]} m \cdot p(j) \cdot 2^j\epsilon \cdot 2^{-(j-1)}\right) + m \cdot 2^{-\ell} \\
&\leq 2m\epsilon \cdot \left(\sum_{j\in[\ell]} p(j)\right) + m \cdot \epsilon/2 \\
&< 2 \cdot \frac{m\epsilon}{2}
\end{aligned}$$

where the last inequality uses $\sum_{j\in[\ell]} p(j) < 1/4$. But this contradicts the hypothesis that $\sum_{i\in[m]} \delta_i \geq m\epsilon$.

We next present the modified ϵ-tester. On input $(x_1, ..., x_m)$, for every $j \in [\ell]$, the tester samples $O((p(j) \cdot 2^j\epsilon)^{-1})$ indices i, and 2^{-j}-tests each input x_i w.r.t Π with error probability $p(j)^2 \cdot 2^j\epsilon/O(1) = \exp(-\Theta(\ell + 1 - j))$. Hence, we make $O(\ell + 1 - j) \cdot \mathtt{Q}_{2^{-j}}(\Pi)$ queries to each x_i that is sampled in the j^{th} iteration, and can neglect the the probability that we obtained a wrong result

[7] Error reduction is used here in order to upper bound the probability that any of the tests returns 0 when $(x_1, ..., x_m) \in \Pi^m$. Indeed, this is not necessary in the case of one-sided error. Actually, one can avoid the error reduction step also in the case of two-sided error by using the ideas underlying Lemma 1.1 (and save a factor of $O(\ell)$), but we did not bother to do so.

for any x_i in any iteration (since a wrong answer is obtained in the j^{th} iteration with probability at most $p(j)/O(1)$). The number of queries made by this tester is upper-bounded by

$$\begin{aligned}\sum_{j\in[\ell]} O((p(j)\cdot 2^j\epsilon)^{-1})\cdot O(\ell+1-j)\cdot \mathtt{Q}_{2^{-j}}(\Pi) &\leq O(1/\epsilon)\cdot\sum_{j\in[\ell]}\frac{\ell+5-j}{p(j)2^j}\cdot\left(\frac{\epsilon/4}{2^{-j}}\right)^c\cdot\mathtt{Q}_{\epsilon/4}(\Pi)\\ &= O(\epsilon^{c-1})\cdot\sum_{k\in[\ell]}(k+4)^3\cdot 2^{(c-1)(\ell+1-k)}\cdot\mathtt{Q}_{\epsilon/4}(\Pi)\\ &\leq O(1)\cdot\sum_{k\in[\ell]}(k+4)^3\cdot 2^{-(c-1)k}\cdot\mathtt{Q}_{\epsilon/4}(\Pi)\end{aligned}$$

where the first inequality uses the theorem's hypothesis, the equality uses the definition of $p(j)$ (and the substitution $k=\ell+1-j$), and the last inequality uses $\ell=\lceil\log_2(1/\epsilon)\rceil+1<\log_2(4/\epsilon)$. Using $\sum_{k\in[\ell]}(k+4)^3\cdot 2^{-c'k}=O(1)$, for any $c'>0$, the claim follows. ■

6 Ramifications: Nonadaptivity and One-Sided Error

In this section, we consider the ramification of our study to nonadaptive algorithms and to one-sided error algorithms. Specifically, in Section 6.1 we consider nonadaptive algorithms (with two-sided error), whereas in Section 6.2 we consider (adaptive) one-sided error algorithms. In each of these cases, we compare the complexity of restricted algorithms solving the multiple-instance problems to the complexity of similarly restricted ϵ-testers of Π.

We note that nonadaptivity "dominates" one-sided error (see Section 6.3): The results for nonadaptive algorithms with one-sided error behave more like the results for nonadaptive algorithms with two-sided error than the results for general (i.e., adaptive) algorithms with one-sided error.

6.1 Nonadaptive algorithms

An algorithm (or rather an oracle machine) is called nonadaptive if it determines its queries solely based on its randomness, regardless of the answers obtained to "prior" queries. (Indeed, in this case, the order of the queries is arbitrary and/or immaterial.)

As noted at the beginning of Section 1.4, the lower bounds for the Direct Sum and Product problems are much easier to establish in the nonadaptive case. In this case, the queries of the algorithm solving the multi-instance problem are oblivious of the instances, and so we can easily derive from it a single-instance algorithm (of easily related query complexity). On the other hand, the efficient reduction of the Direct Product Problem to the Direct Sum Problem, captured by Lemma 1.1, does not seem to work anymore (since the reduction that we presented is inherently adaptive). Letting $\mathtt{Q}^{\text{na}}_\epsilon$ denote the minimum query complexity of all nonadaptive ϵ-testers for Π, we get:

Theorem 6.1 (nonadaptive query complexity of multiple-instance problems): *For every property Π, proximity parameter ϵ, and integer m, the following holds.*

The Direct Sum Problem: *The nonadaptive query complexity of $\mathtt{DS}^m_\epsilon(\Pi)$ is $\Theta(m\cdot\mathtt{Q}^{\text{na}}_\epsilon(\Pi))$.*

The Direct Product Problem: *The nonadaptive query complexity of $\mathtt{DP}^m_\epsilon(\Pi)$ is $\Omega(m)\cdot\mathtt{Q}^{\text{na}}_\epsilon(\Pi)$ and $O(m\log m)\cdot\mathtt{Q}^{\text{na}}_\epsilon(\Pi)$.*

The Concatenation Problem: *For $c \geq 1$, suppose that $\mathtt{Q}_\epsilon(\Pi)$ increases at least linearly with $1/\epsilon^c$. Then, the nonadaptive query complexity of $\mathtt{CP}^m_\epsilon(\Pi)$ is $O(\log(1/\epsilon))^3 \cdot \mathtt{Q}^{\mathrm{na}}_{\epsilon/4}(\Pi)$ if $c = 1$, and $O(\mathtt{Q}^{\mathrm{na}}_{\epsilon/4}(\Pi))$ otherwise* (i.e., for $c > 1$).

Indeed, closing the gap for the Direct Product Problem is left as an open problem. As observed by Ron Rothblum, the gap disappears whenever $\mathtt{Q}^{\mathrm{na}}_\epsilon(\Pi) = O(\mathtt{Q}^{\mathrm{dr}}_\epsilon(\Pi))$, where $\mathtt{Q}^{\mathrm{dr}}_\epsilon(\Pi)$ is as defined in Section 6.3. Note that it may be the case that the nonadaptive query complexity of $\mathtt{DP}^m_\epsilon(\Pi)$ is $\Theta(m) \cdot \mathtt{Q}^{\mathrm{na}}_\epsilon(\Pi)$ for some Π and $\Theta(m \log m) \cdot \mathtt{Q}^{\mathrm{na}}_\epsilon(\Pi)$ for others.

Proof Sketch: Both lower bounds follow by starting with an algorithm that solves the multi-instance problem with query complexity q, and deriving an ϵ-tester for Π with *expected* query complexity of at most q/m. By truncating executions that make more than $10q/m$ queries, we obtain an ϵ-tester that errs with probability $(1/3) + 0.1$. Lastly, error reduction yields a standard ϵ-tester for Π of query complexity $O(q/m) \geq \mathtt{Q}_\epsilon(\Pi)$, and the lower bound claims follow (for both problems). Regarding the Concatenation Problem, we note that the reductions presented in Section 5 are actually nonadaptive, and so the upper bound claims follow. ■

6.2 One-sided error algorithms

An ϵ-tester for Π is said to have one-sided error if it always accepts (i.e., output 1) inputs in Π (rather than accept them with probability at least 2/3). We denote by $\mathtt{Q}^{\mathrm{ose}}_\epsilon$ the minimum query complexity of all one-sided error ϵ-testers for Π. The notion of one-sided error algorithms extends naturally to the three multiple-instance problems we have studied: In the Direct Sum Problem it asserts that σ_i is always 1 if $x_i \in \Pi$, whereas in the other two problems it asserts that the algorithm always accepts a sequence in Π^m.

In the context of one-sided error algorithms, there is a simpler reduction of the Direct Product Problem to the Direct Sum Problem (i.e., simpler than the one captured by Lemma 1.1); that is, solving $\mathtt{DP}^m_\epsilon(\Pi)$ (with one-sided error) reduces to a single invocation of a (one-sided error) algorithm solving $\mathtt{DS}^m_\epsilon(\Pi)$. The Direct Sum solver outputs 1 if and only if the m-bit long vector of answers obtained for the Direct Sum Problem is all 1. This implies that *the one-sided error query complexity of $\mathtt{DP}^m_\epsilon(\Pi)$ is $O(m \cdot \mathtt{Q}^{\mathrm{ose}}_\epsilon(\Pi))$*, but we shall see a stronger result below. Regarding the Concatenation Problem, as in the nonadaptive case, it is clear that the algorithms presented in Section 5 operate also in the case of one-sided error (i.e., the reductions preserve one-sided error). Actually, we can save a $\log(1/\epsilon)$ factor (see Footnote 7).

Regarding the lower bounds, it turns out that the argument for the Direct Sum Problem can be adapted (as shown below), but the one for Direct Product Problem fails. In fact, there are cases in which the one-sided error query complexity of the Direct Product Problem is very close to the one-sided error query complexity of testing Π. Specifically, the one-sided error query complexity of $\mathtt{DP}^m_\epsilon(\Pi)$ is $O(m \cdot \mathtt{Q}_\epsilon(\Pi) + \mathtt{Q}^{\mathrm{ose}}_\epsilon(\Pi))$, whereas in some cases $\mathtt{Q}_\epsilon(\Pi)$ is much smaller than $\mathtt{Q}^{\mathrm{ose}}_\epsilon(\Pi)$ (e.g., ϵ-testing ρ-clique in the dense graphs model has poly$(1/\epsilon)$-query two-sided error tester [5, Sec. 7], but no $o(\sqrt{n})$-query one-sided error testers [5, sec. 10.1.6]).[8]

Theorem 6.2 (one-sided error query complexity of multiple-instance problems): *For every property Π, proximity parameter ϵ, and integer m, the following holds.*

The Direct Sum Problem: *The one-sided error query complexity of $\mathtt{DS}^m_\epsilon(\Pi)$ is $\Theta(m \cdot \mathtt{Q}^{\mathrm{ose}}_\epsilon(\Pi))$.*

[8] Another natural example is ϵ-testing cycle-freeness in the bounded-degree graphs model [8, Sec. 4]. Also note that the set Π of n-bit strings having at least $n/2$ one-entries has a two-sided error $O(1/\epsilon^2)$-query tester, but requires at least $n/2$ queries for one-sided error testing.

The Direct Product Problem: *The one-sided error query complexity of* $\mathtt{DP}^m_\epsilon(\Pi)$ *is* $\Theta(m \cdot \mathtt{Q}_\epsilon(\Pi) + \mathtt{Q}^{\mathrm{ose}}_\epsilon(\Pi))$.

The Concatenation Problem: *For* $c \geq 1$, *suppose that* $\mathtt{Q}_\epsilon(\Pi)$ *increases at least linearly with* $1/\epsilon^c$. *Then, the one-sided error query complexity of* $\mathtt{CP}^m_\epsilon(\Pi)$ *is* $O(\log(1/\epsilon))^2 \cdot \mathtt{Q}^{\mathrm{ose}}_{\epsilon/4}(\Pi)$ *if* $c = 1$, *and* $O(\mathtt{Q}^{\mathrm{ose}}_{\epsilon/4}(\Pi))$ *otherwise* (i.e., for $c > 1$).

Proof Sketch: When proving the lower bound for $\mathtt{DS}^m_\epsilon(\Pi)$, we modify the distribution D as follows. For $q = \mathtt{Q}^{\mathrm{ose}}_\epsilon(\Pi)$, the algorithmic player we consider here selects (randomly or deterministically) a deterministic oracle machine that makes at most $q-1$ queries and *always outputs 1 on any input in* Π. As before, the adversarial player selects (randomly or deterministically) an input in $\Pi \cup \overline{\Gamma}_\epsilon(\Pi)$. Now, D is a distribution on inputs in $\Pi \cup \overline{\Gamma}_\epsilon(\Pi)$ such that for any deterministic machine M of query complexity at most $q-1$ that always accepts inputs in Π it holds that $\Pr_{x \in D}[M^x = 0 | x \in \overline{\Gamma}_\epsilon(\Pi)] < 2/3$ (which in particular means that $\Pr_{x \in D}[x \in \overline{\Gamma}_\epsilon(\Pi)] > 0$). Finally, we proceed as in the proof of Theorem 3.1, with two exceptions:

1. Here we start with a one-sided error machine $\overline{M_0}$ that solves the multi-instance problem $\mathtt{DS}^m_\epsilon(\Pi)$, and our aim is to derive a one-sided error ϵ-tester for Π.

2. When truncating executions that make too many queries to the i^{th} instance, we make the modified algorithm output 1.

   Note that in the proof of Theorem 3.1, the output in these truncated executions was left unspecified. Hence, the specific setting used above does not affect the validity of the analysis of the case that the instance is in $\overline{\Gamma}_\epsilon(\Pi)$. But this specific setting guarantees that inputs in Π are always accepted (by the single-instance algorithm that we derive).

This completes the proof of the lower bound for $\mathtt{DS}^m_\epsilon(\Pi)$.

Turning to $\mathtt{DP}^m_\epsilon(\Pi)$, we first note that the lower bound follows by combing the lower bound on two-sided testers (i.e., $\Omega(m \cdot \mathtt{Q}_\epsilon(\Pi))$) with the obvious lower bound of $\mathtt{Q}^{\mathrm{ose}}_\epsilon(\Pi)$. As for the upper bound, we consider an algorithm that first finds a candidate instance in $\overline{\Gamma}_\epsilon(\Pi)$, by using a two-sided error ϵ-tester, and then applies a one-sided error ϵ-tester to it. The first step is easy to implement by invoking the two-sided error tester $O(\log m)$ times on each instance (and ruling by majority), but we aim at a better upper bound. The idea is to use a modification of the algorithm presented in the proof of Lemma 1.1. Specifically, on input $(x_1, ..., x_m)$, the algorithm, denoted A, proceeds as follows:

1. Algorithm A invokes the reduction presented in the proof of Lemma 1.1, while implementing a two-sided error algorithm for the Direct Product Problem (of Π) by using the two-sided error ϵ-tester for Π. If the reduction outputs 1, then A outputs 1. Otherwise, let I be the "too big" set considered at the iteration in which the reduction halts with output 0.

2. Algorithm A selects uniformly at random $i \in I$, invokes the one-sided error ϵ-tester for Π on x_i, and outputs the output it has obtained.

By Lemma 1.1, algorithm A has query complexity $O(m) \cdot \mathtt{Q}_\epsilon(\Pi) + \mathtt{Q}^{\mathrm{ose}}_\epsilon(\Pi)$, and by its construction it only outputs 0 if the one-sided error tester of Π outputs 0 on one of the x_i's. Thus, A has one-sided error. On the other hand, if some of the x_i's are in $\overline{\Gamma}_\epsilon(\Pi)$, then with probability at least $2/3$, algorithm A proceeds to its second step. A closer look at the proof of Lemma 1.1 reveals that we can guarantee that, in this case, with probability at least $3/4$, at least half of the instances in I are

in $\overline{\Gamma}_\epsilon(\Pi)$. (All that is needed is to reduce the error probability on individual instances from $2^{-(j+3)}$ to $2^{-(j+4)}$.) This is the case because, for $j \geq 2$, with probability at least 3/4, at the beginning of the j^{th} iteration the set I may contain at most $0.5 \cdot 2^{-(j-1)} \cdot m$ instances in Π, whereas halting occurs when $|I| > 2^{-(j-1)} \cdot m$. Hence, with probability at least $\frac{3}{4} \cdot 0.5 \cdot \frac{2}{3}$, algorithm A outputs 0. ■

6.3 Nonadaptive algorithms with one-sided error

The proof of Theorem 6.1 can be adapted to the case of (nonadaptive) algorithms of one-sided error. All that is required is to be careful about the lower bound arguments. Specifically, when truncating executions that make too many queries to a certain instance, we let the algorithm output 1 (rather than halt with arbitrary output). Actually, we can close the gap left in Theorem 6.1 (and get a tight result also for the Direct Product Problem) by using the simple reduction of the Direct Product Problem to the Direct Sum Problem mentioned in Section 6.2. Hence, letting $\mathtt{Q}^{\mathrm{dr}}_\epsilon$ denote the minimum query complexity of all nonadaptive one-sided error ϵ-testers for Π, where "dr" stands for doubly restricted, we get:

Theorem 6.3 (doubly restricted query complexity of multiple-instance problems): *For every property Π, proximity parameter ϵ, and integer m, the following holds.*

The Direct Sum Problem: *The nonadaptive one-sided error query complexity of $\mathtt{DS}^m_\epsilon(\Pi)$ is $\Theta(m \cdot \mathtt{Q}^{\mathrm{dr}}_\epsilon(\Pi))$.*

The Direct Product Problem: *The nonadaptive one-sided error query complexity of $\mathtt{DP}^m_\epsilon(\Pi)$ is $\Theta(m \cdot \mathtt{Q}^{\mathrm{dr}}_\epsilon(\Pi))$.*

The Concatenation Problem: *For $c \geq 1$, suppose that $\mathtt{Q}_\epsilon(\Pi)$ increases at least linearly with $1/\epsilon^c$. Then, the nonadaptive one-sided error query complexity of $\mathtt{CP}^m_\epsilon(\Pi)$ is $O(\log(1/\epsilon))^2 \cdot \mathtt{Q}^{\mathrm{dr}}_{\epsilon/4}(\Pi)$ if $c = 1$, and $O(\mathtt{Q}^{\mathrm{dr}}_{\epsilon/4}(\Pi))$ otherwise* (i.e., for $c > 1$).

Acknowledgments

We are grateful to Tom Gur and Ron Rothblum for comments on an early draft of this paper.

We thank Ilan Newman for calling our attention to the fact that Lemma 1.1 is implicit in the work of Feige *et al.* [1].

References

[1] U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with Noisy Information. *SIAM Journal on Computing*, Vol. 23 (5), pages 1001–1018, 1994.

[2] O. Goldreich. *Foundations of Cryptography – Basic Tools*. Cambridge University Press, 2001.

[3] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[4] O. Goldreich, editor. *Property Testing – Current Research and Surveys*. Lecture Notes in Computer Science, Vol. 6390, Springer, 2010.

[5] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, pages 653–750, July 1998. Extended abstract in *37th FOCS*, 1996.

[6] O. Goldreich and L.A. Levin. A hard-core predicate for all one-way functions. In the proceedings of *21st ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[7] O. Goldreich, N. Nisan and A. Wigderson. On Yao's XOR-Lemma. *ECCC*, TR95-050, 1995.

[8] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, pages 302–343, 2002. Extended abstract in *29th STOC*, 1997.

[9] L.A. Levin. One-way functions and pseudorandom generators. In proc. of the *17th ACM Symposium on the Theory of Computing*, pages 363–365, 1985.

[10] I. Newman. Computing in Fault Tolerant Broadcast Networks and Noisy Decision Trees. *Random Struct. Algorithms*, Vol. 34 (4), pages 478–501, 2009.

[11] D. Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, Vol. 1 (3), pages 307–402, 2008.

[12] D. Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in TCS*, Vol. 5 (2), pages 73–205, 2009.

[13] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2), pages 252–271, 1996.

[14] J. von Neumann. Various techniques used in connection with random digits. *Applied Math Series*, Vol. 12, pages 36–38, 1951. Reprinted in *von Neumann's Collected Works*, Vol. 5, pages 768–770, Pergamon, 1963.

[15] A.C.C. Yao. Lower Bounds by Probabilistic Arguments (Extended Abstract). In *Proc. 24th IEEE Symposium on Foundations of Computer Science*, pages 420–428, 1983.

Appendix: In Passing

In this appendix, we elaborate on two comments that were made in the main text. Section A.1 discusses a well-known result, while expressing an opinion of its contents. Section A.2 explicitly presents a useful idea, which is implicit in many prior works (as well as in Section 5).

A.1 On Yao's MiniMax Principle

As is well known, Yao [15] made the influential observation that von Neumann's MiniMax Principle for zero-sum games [14] can be applied in the context of the analysis of randomized algorithms. In particular, Yao considered an "algorithmic" player selecting a distribution over deterministic algorithms and an "adversarial (input)" player selecting an input distribution. Then, the application of the game theoretic principle implies that the profit (e.g., success probability) of the best randomized algorithm on the worst-case input equals the upper bound on the profit of any (deterministic) algorithm under some fixed distribution of inputs. (The same applies when one consider the cost (e.g., in resources) incurred by the algorithm.)

Our point is that the above statement has two directions. The first direction (which is obvious and more commonly used) is that the profit of the best algorithm on the worst-case input cannot exceed the profit any algorithm may get under some (adversarially chosen) input distribution. The second direction, which is the remarkable one, asserts that the upper bound obtained in this way (i.e., by a worst adversarial selection of an input distribution) is tight. In other words, the foregoing method of bounding the profit of randomized algorithms is complete (or optimal). Thus, in our opinion, Yao contribution was in pioneering and advocating the method of obtaining lower bounds via the design of adversarial input distributions, not in proving the soundness of this method. (Again, the method's soundness is obvious, what is remarkable is its completeness.)

For sake of clarity, let us cast the above discussion in precise terms. Let $p(r,c)$ be the profit attained by the row player, when it picks the row r and the column player picks c. Let V be the value obtained when the row player picks r under the best possible distribution on rows, denoted $\mathcal{R}$, and the column player picks c to minimize the row player's profit; that is, $V \stackrel{\text{def}}{=} \min_c\{\mathrm{E}_{r\leftarrow\mathcal{R}}[p(r,c)]\}$. Let U be the upper bound on the profit obtained when the column player uses a distribution, denoted $\mathcal{C}$, that minimizes the profit of the best choice of a row r; that is, $U \stackrel{\text{def}}{=} \max_r\{\mathrm{E}_{c\leftarrow\mathcal{C}}[p(r,c)]\}$. Obviously, $V \leq U$:

$$\begin{aligned} V &= \min_c\{\mathrm{E}_{r\leftarrow\mathcal{R}}[p(r,c)]\} \\ &\leq \mathrm{E}_{r\leftarrow\mathcal{R},c\leftarrow\mathcal{C}}[p(r,c)] \\ &\leq \max_r\{\mathrm{E}_{c\leftarrow\mathcal{C}}[p(r,c)]\} \\ &= U. \end{aligned}$$

The actual contents of the MiniMax Principle is that this upper bound is actually tight; that is, $V = U$. The known proofs of the latter assertion are far more complex than the above manipulation: The classical proof proceeds by presenting a transformation over the space of strategy pairs such that its fixed-points are equilibria pairs, and applying Brouwer's fixed-point theorem. A popular alternative proof proceeds by formulating the problem of finding an optimal strategy (for the row player) as a linear program and applying the strong LP-Duality Theorem.

Summary. We claim that three things are being confused: (1) The generic fact that $V \leq U$; (2) the generic fact that V actually equals U; and (3) the suggestion that in many specific settings

(e.g., where the payoff represents the success probability of an algorithm on an input) it is beneficial to prove upper bounds on V by proving upper bounds on U.

A.2 On Levin's economical work investment strategy

In some situations one can sample a huge space that contains elements of different quality such that elements of lower quality require more work to utilize. The aim is to utilize some element, but the work required for utilizing the various elements is not known a priori, and it only becomes known after the entire amount of required work is invested. Note that it may be that most of the elements are of very poor quality, and so it is not a good idea to select a single element and invest as much work as is needed to utilize it. Instead one may want to select many samples points and invest in each of them a limited amount of work (which may be viewed as probing the required amount of work).

To be more concrete, suppose that the work that needs to be invested in a sample point s (in order to utilize it) is inversely proportional to its (unknown to us) quality $q(s)$. We only know a lower bound ϵ on the average quality of an element (i.e., $\mathrm{E}_s[q(s)] > \epsilon$), and we wish to minimize the total amount of work invested in utilizing some element. One natural strategy that comes to mind is to sample $O(1/\epsilon)$ points and invest $O(1/\epsilon)$ work in each of these points. In this case we succeed with constant probability, while investing $O(1/\epsilon^2)$ work. The analysis is based on the fact that $\mathrm{E}_s[q(s)] > \epsilon$ implies that $\Pr_s[q(s) > \epsilon/2] > \epsilon/2$. The following fact suggests a more economical strategy.

Fact A.1 (a refined counting argument): *Let $\mathcal{D}$ be a probability distribution, $q : \mathrm{Supp}(\mathcal{D}) \to [0,1]$, and $\epsilon \in (0,1]$. Suppose that $\mathrm{E}_{s \leftarrow \mathcal{D}}[q(s)] > \epsilon$, and let $\ell = \lceil \log_2(2/\epsilon) \rceil$. Then, there exists $j \in [\ell]$ such that $\Pr_{s \leftarrow \mathcal{D}}[q(s) > 2^{-j}] > 2^j \epsilon / 4\ell$.*

Hence, an alternative strategy can succeed, with constant probability, by investing $\widetilde{O}(1/\epsilon)$ work. Specifically, for each $j \in [\ell]$, we take $O(\ell/2^j\epsilon)$ samples, and invest $O(2^j)$ work in each of these sample points. (Note that we cannot expect to invest less than $o(1/\epsilon)$ work in total, since we may have $q(s) = 2\epsilon$ for each sample point s, and so the alternative strategy is almost optimal.)

We learned this alternative strategy from Leonid Levin in the mid-1980s. This strategy is used in [9] (see the last paragraph of [9, Sec. 9]) and is stated explicitly in [6, Lem. 3] (see [2, Clm. 2.5.4.1] for an alternative presentation). Within the context of property testing, this strategy was first used in [8] (see Lemma 3.3 in the proceeding version and Lemma 3.6 in the journal version).

Proof: Let $B_j \stackrel{\mathrm{def}}{=} \{s : 2^{-j} < q(s) \leq 2^{-(j-1)}\}$, and assume towards the contradiction that for every $j \in [\ell]$ it holds that $\Pr_s[q(s) > 2^{-j}] \leq 2^j \epsilon / 4\ell$. This implies that for every $j \in [\ell]$ it holds that $\Pr_s[s \in B_j] \leq 2^j \epsilon / 4\ell$ (and $q(s) \leq \epsilon/2$ for every $s \notin \bigcup_{j \in [\ell]} B_j$). We get

$$\begin{aligned}
\mathrm{E}_s[q(s)] &\leq \frac{\epsilon}{2} + \sum_{j \in [\ell]} \Pr_s[s \in B_j] \cdot 2^{-(j-1)} \\
&\leq \frac{\epsilon}{2} + \sum_{j \in [\ell]} \frac{2^j \epsilon}{4\ell} \cdot 2^{-(j-1)} \\
&= \frac{\epsilon}{2} + \sum_{j \in [\ell]} \frac{\epsilon}{2\ell}
\end{aligned}$$

which contradicts the fact's hypothesis. ■

The case of required work that increases faster than $O(1/q(\cdot))$. The above description refers to the case that the work that needs to be invested in utilizing an element is inversely proportional to its quality (i.e., the work that needs to be invested in s is $\Theta(1/q(s))$). In that case, we sought a set S such that the product $\Pr_s[s \in S] \cdot \min_{s \in S}\{q(s)\}$ is maximized. (Actually, we identified $O(\log(1/\epsilon))$ candidate sets S, and invested $O(1/\epsilon)$ work in each of them.) We now consider the case that for some $c > 1$ (e.g., $c = 2$), the work that needs to be invested in order to utilize the element s is $\Theta(1/q(s)^c)$. In this case, we seek a set S such that the product $\Pr_s[s \in S] \cdot \min_{s \in S}\{q(s)^c\}$ is maximized. (Indeed, the case of $c = 2$ arises quite often in applications; for example, it arises whenever one wishes to approximate some $[0,1]$-valued quantity up to $\pm q(s)$.)

Here the straightforward solution is to sample $O(1/\epsilon)$ points and invest $O(1/\epsilon^c)$ work in each of these sample points. In this case we succeed with constant probability, while investing $O(1/\epsilon^{c+1})$ work. The following fact suggests a more economical procedure.

Fact A.2 (an alternatively refined counting argument): *Let $\mathcal{D}$ be a probability distribution, $q : \mathrm{Supp}(\mathcal{D}) \to [0,1]$, and $\epsilon \in (0,1]$. Suppose that $\mathrm{E}_{s \leftarrow \mathcal{D}}[q(s)] > \epsilon$, and let $\ell = \lceil \log_2(2/\epsilon) \rceil$. Then, there exists $j \in [\ell]$ such that $\Pr_{s \leftarrow \mathcal{D}}[q(s) > 2^{-j}] > 2^j \epsilon/(\ell + 5 - j)^2$.*

Hence, an alternative strategy can succeed with constant probability by investing $O(1/\epsilon^c)$ work, which is optimal (since we may have $q(s) = 2\epsilon$ for all s). Specifically, for each $j \in [\ell]$, we take $O((\ell + 5 - j)^2/2^j\epsilon)$ samples, and invest $O(2^{cj})$ work in each sample point. Indeed, for every $c > 1$, it holds that

$$\begin{aligned}\sum_{j \in [\ell]} \frac{(\ell + 5 - j)^2}{2^j \epsilon} \cdot 2^{cj} &= \frac{1}{\epsilon} \cdot \sum_{k \in [\ell]} (k+4)^2 \cdot 2^{(c-1)\cdot(\ell+1-k)} \\ &= O(1/\epsilon)^c \cdot \sum_{k \in [\ell]} (k+4)^2 \cdot 2^{-(c-1)\cdot k}\end{aligned}$$

which equals $O(1/\epsilon^c)$, because $\sum_{k \in [\ell]} (k+4)^2 \cdot 2^{-c'k} = O(1)$ for every $c' > 0$.

Proof: The proof is by a simple adaptation of the proof of Fact A.1. As before, let $B_j \stackrel{\text{def}}{=} \{s : 2^{-j} < q(s) \leq 2^{-(j-1)}\}$, and assume towards the contradiction that for every $j \in [\ell]$ it holds that $\Pr_s[q(s) > 2^{-j}] \leq 2^j \epsilon/(\ell + 5 - j)^2$. This implies that for every $j \in [\ell]$ it holds that $\Pr_s[s \in B_j] \leq 2^j \epsilon/(\ell + 5 - j)^2$ (and $q(s) \leq \epsilon/2$ for every $s \notin \bigcup_{j \in [\ell]} B_j$). We get

$$\begin{aligned}\mathrm{E}_s[q(s)] &\leq \frac{\epsilon}{2} + \sum_{j \in [\ell]} \Pr_s[s \in B_j] \cdot 2^{-(j-1)} \\ &\leq \frac{\epsilon}{2} + \sum_{j \in [\ell]} \frac{2^j \epsilon}{(\ell + 5 - j)^2} \cdot 2^{-(j-1)} \\ &= \frac{\epsilon}{2} + \sum_{k \in [\ell]} \frac{2\epsilon}{(k+4)^2}\end{aligned}$$

which contradicts the fact's hypothesis (since $\sum_{k \in [\ell]} (k+4)^{-2} < 1/4$). ■