Property Testing of LP-Type Problems

Rogers Epstein
Massachusetts Institute of Technology, Cambridge, MA, USA
rogersep@mit.edu

Sandeep Silwal
Massachusetts Institute of Technology, Cambridge, MA, USA
silwal@mit.edu

Abstract

Given query access to a set of constraints S, we wish to quickly check if some objective function φ subject to these constraints is at most a given value k. We approach this problem using the framework of property testing where our goal is to distinguish the case $\varphi(S) \leq k$ from the case that at least an ϵ fraction of the constraints in S need to be removed for $\varphi(S) \leq k$ to hold. We restrict our attention to the case where (S, φ) are LP-Type problems which is a rich family of combinatorial optimization problems with an inherent geometric structure. By utilizing a simple sampling procedure which has been used previously to study these problems, we are able to create property testers for any LP-Type problem whose query complexities are independent of the number of constraints. To the best of our knowledge, this is the first work that connects the area of LP-Type problems and property testing in a systematic way. Among our results are property testers for a variety of LP-Type problems that are new and also problems that have been studied previously such as a tight upper bound on the query complexity of testing clusterability with one cluster considered by Alon, Dar, Parnas, and Ron (FOCS 2000). We also supply a corresponding tight lower bound for this problem and other LP-Type problems using geometric constructions.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases property pesting, LP-Type problems, random sampling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.98

Category Track A: Algorithms, Complexity and Games

Funding *Sandeep Silwal*: Research supported by the NSF Graduate Research Fellowship under Grant No. 1122374.

Acknowledgements We would like to thank Ronitt Rubinfeld, Piotr Indyk, Bertie Ancona, and Rikhav Shah for helpful feedback.

1 Introduction

Many problems in combinatorial optimization can be represented as a pair (S, φ) where S is a set of constraints and φ is a function of the constraints that we would like to minimize. This class includes many problems that are NP-hard, even for the decision version of some problems where we would like to determine if $\varphi(S)$ is at most some constant k. For instance, let S be constraints that say two nodes in a graph are connected by an edge (so S can be thought of as a set of edges) and φ be the chromatic number of a graph with those edges. Then it is NP-complete to determine if $\varphi(S) \leq 3$.

In this work, we consider a relaxation of the above hard class of problems by using the framework of *property testing*. Specifically, given a value parameter k and distance parameter ϵ, we wish to determine if $\varphi(S) \leq k$ or if (S, φ) is ϵ-far from $\varphi(S) \leq k$, where ϵ-far means that at least $\epsilon|S|$ many of the constraints of S need to be removed for $\varphi(S) \leq k$ to hold. We assume we have query access to the constraints and knowledge of φ and our goal is to perform property testing while minimizing the number of constraints of S we access.

Even under the property testing setting, this question is too broad. Therefore, we focus our attention to *LP-Type Problems*, formally described in Definition 1. Informally, these problems have an underlying geometrical structure, much like that of linear programs, which can be used to create efficient testing algorithms.

Our main result is a general algorithm that is able to perform property testing for *any* LP-Type problem with $O(\delta/\epsilon)$ queries where δ is the dimension of the LP-Type problem, formally defined in Section 2. In many cases, this bound is tight such as testing clusterability using one cluster considered in [1]. To the best of our knowledge, this is the first work that connects the area of LP-Type problems and property testing in a systematic way. The class of these problems is quite general and includes problems which have been previously studied individually in property testing, such as testing clusterability of points in [1], and newer testing problems, such as determining if a set of linear constraints is feasible or "far" from feasible. We also give a matching lower bound for many of these problems using geometric constructions. For a comprehensive overview of our contributions, see Section 2.2.

1.1 Related Work

Many problems in property testing can be modeled as a set of constraints and some optimization function over these constraints. These include well studied graph problems such as bipartiteness, expansion, k-colorability, and many other problems [14, 15, 16, 23]. This line of work was initiated by Goldreich and Ron and there are many results in the area of graph property testing. For more information about graph property testing, see [13] and the references within. Overall, these testing problems differ from our setting where our queries are essentially access to random constraints.

This model where queries are accesses to constraints have also been studied in the case where we wish to test properties of a metric space and queries are access to points (see [19, 1, 21]). There are instances of these problems that are also examples of LP-Type problems that we consider. For more information, see Section 2.2.

LP-Type problems have a rich literature and there have been many previous work on them, including general algorithms to solve LP-Type problems [25, 6, 26, 18]. The algorithms for these problems have runtimes that are generally linear in the number of constraints, but exponential in the dimension of the LP-Type problem (see Definition 3). This is in contrast to our testing algorithms that have no dependence on the number of constraints.

Furthermore, many properties of LP-Type problems have been generalized to a larger class of problems called violator spaces [11, 4]. We do not explicitly consider them here since these problems do not yield any additional interesting property testing applications but our results carry over to this setting in a straightforward manner.

There are previous works on geometric property testing, for example, testing convexity of point sets [7, 20], testing disjointness of geometric bodies [9], and testing properties of two dimensional images [22, 24, 2, 3]. While many of these work share the paradigm of "sample few objects and test" that is common among many property testers, including ours, the problems considered in these works are very different than the problems we focus on (see 2.2).

There is also existing work on property testing for constraint satisfaction problems (CSPs) where given an instance of a CSP, one is given query access to an assignment of the variables and the task is to determine if the assignment is "close" or "far" from satisfying the instance [5]. This is different than our setting where we wish to check if $\varphi(S) \leq k$ where φ is a function of the constraints in S.

The paper of Czumaj and Sohler cited in [8] is closest in spirit to our work, as it also introduces a framework for property testing for a general class of problems which they refer to as *abstract combinatorial problems* (ACPs). This class of problems are similar to LP-type problems as they both include a set of constraints and consider a restricted set of subsets called a basis (see Section 2 for a formal definition of LP-Type problems). Furthermore, [8] exploit the combinatorial structure of ACPs to develop testing algorithms, just as we do for LP-Type problems. However, there are some key differences between ACPs and LP-Type problems, such as the fact that LP-Type problems are equipped with a function ϕ that is optimized over a set of constraints. LP-Type problems also have a clean correspondence between dimension and the number of "violators" for a particular basis as formulated in Lemma 11 and Corollary 13, which lets us automatically translate between the dimension and the property testing query complexity.

In terms of concrete property testing problems, [8] also consider the smallest enclosing ball problem (see Section 2.2). We achieve a better query complexity bound than [8] for this problem, but it is important to note that the results of [8] also hold for the general problem of clustering with multiple balls while we only consider one ball. However, the rest of the problems considered in [8] do not overlap with the problems we present in this paper and furthermore, the problems we study are inherently geometric in nature. Lastly, while ACPs are the invention of the authors in [8], there exists rich literature for LP-Type problems outside of the domain of property testing.

Lastly, we note that using a few samples to determine information about your underlying data has also been studied in the PAC-learning literature. In that setting, the goal is to learn a classifier of your dataset and recently, the question of optimal sample complexity for PAC learning has been settled, see for instance [17]. The key differences between the PAC learning setting and our work is that our "datapoints" do not necessarily come with a label. For instance, the PAC learning framework would be valid in the example where we have labeled points in $\mathbb{R}^d$ and we wish to learn a separator from a family class such as balls. However, this framework would not apply if we wished to compute the smallest enclosing ball of the points. We do consider one instance where a PAC learning framework would apply and that is problem of property testing if a set of labeled points are linearly separable or far from it (for more information, see Section 2.2). In this example, a valid approach would be to use the PAC learning framework and learn a linear classifier. Instead of pursuing this route, we are able to cast this problem as an example of a LP-Type problem and achieve the optimal sample complexity using the theory we develop.

1.2 Organization

In Section 2, we formally define the class of LP-Type problems. In Section 2.2 we outline our contributions. In Section 3, we present our algorithms and prove their correctness and in Section 4, we apply our algorithm to specific LP-Type problems. Finally, we complement some of these problems with lower bounds in Section 5.

2 Preliminaries

2.1 Notation and Definitions

We define LP-Type problems as well as some related concepts. These definitions are standard in the literature for LP-Type problems but we reproduce them below for the sake of completeness. For more information, see [26, 18, 12].

▶ **Definition 1** (LP-Type Problem). *Let S be a finite set and φ be a function that maps subsets of S to some value. We say (S, φ) is a LP-Type problem if φ satisfies the following two properties:*

- *Monotonicity: if $A \subseteq B \subseteq S$ then $\varphi(A) \leq \varphi(B)$*
- *Locality: For all $A \subseteq B \subseteq S$ and elements $x \in S$, if $\varphi(A) = \varphi(B) = \varphi(A \cup \{x\})$, then $\varphi(A) = \varphi(B \cup \{x\})$.*

In general, one should think of S as a set of constraints, and φ as some objective function we wish to minimize (or equivalently, maximize) over these constraints. The canonical example of a LP-Type problem is a linear program where S is a set of linear constraints and φ is a linear functional.

Just as linear programs can be associated with a dimension, which is the number of variables, LP-Type problems in general also have a natural definition of dimension which influences the runtime of many algorithms for LP-Type problems as well as our algorithm for property testing. First, we must define the notion of a basis.

▶ **Definition 2** (Basis of LP-Type problems). *Given an LP-Type problem, a* basis *$B \subseteq S$ is a set such that for all proper subsets $B' \subset B$, we have $\varphi(B') < \varphi(B)$.*

Given the above definitions, we can now define the dimension of a LP-Type problem.

▶ **Definition 3** (Dimension of LP-Type problem). *The* dimension *δ of an LP-Type problem is the largest possible size of a basis $B \subseteq S$. This is sometimes also called the* combinatorial dimension.

There are many examples of well-studied LP-Type problems and in many of these cases explicit bounds, if not exact values, are known regarding their dimensions. For more information, see our contributions in Section 2.2. In general, the dimension of the problem tends to grow with the "difficulty" of solving it and for property testing, our query complexity bound is also a function of the dimension. We now formally define property testing for LP-Type problems.

▶ **Definition 4** (Property Testing of LP-Type problems). *Given an LP-Type problem (S, φ), a parameter k, a distance parameter ϵ, and query access to the constraints in S, we wish to distinguish the following two cases:*

- *Output accept with probability at least $2/3$ if $\varphi(S) \leq k$ (Completeness Case)*
- *Output reject with probability at least $2/3$ if at least $\epsilon|S|$ constraints need to be removed from S for $\varphi(S) \leq k$ to hold (Soundness Case).*

▶ Remark 5. We say that S is ϵ-far if it falls in the soundness case.

2.2 Our Contributions

The main contribution of this paper is a comprehensive algorithm for property testing of LP-Type problems with query complexity $O(\delta/\epsilon)$ where δ is the dimension of the LP-Type problem. Note that this bound is independent of the number of constraints which is $|S|$. Our algorithm is simple and proceeds by first sampling a small set of random constraints in S, constructing a partial solution, and "testing" this partial solution against few other randomly chosen constraints. The analysis that we reject in the ϵ-far case (soundness) is straightforward. However, the main technical challenge lies in showing that our algorithm accepts in the completeness case. To do so, we use a "sampling" lemma which roughly says that for a randomly chosen subset R of S of a particular size (depending on the dimension δ) and x a randomly chosen element of $S \setminus R$, we have $\varphi(R) = \varphi(R \cup \{x\})$. Using this result, we show that we are likely to accept in the completeness case. For the full detailed analysis, see Section 3. All of our algorithms have two-sided error.

We highlight the power of our approach by considering the query complexity bounds that we get for a few selected problems. In many cases, we are also able to get matching lower bounds. More specifically, we obtain the following results as an application of our framework:

1. We consider the problem of determining if a set of linear inequalities in d variables is feasible (there exists a satisfying assignment) or if at least ϵ-fraction of the constraints need to be removed for the set of constraints to be feasible. While this problem does not exactly fall under the LP-Type definition (since there is no optimization function), we modify our general algorithm slightly to given an algorithm with query complexity is $O(d/\epsilon)$. We also modify our approach for this problem to give a *tolerant* tester for linear program feasibility. This passes programs which are ϵ/c-close to feasible for some fixed constant $c > 1$, and rejects those which are ϵ-far from feasible. This tester also has query complexity $O(d/\epsilon)$
2. We study the problem of determining if a set of points in d dimensions labeled $\{+1, -1\}$ is linearly separable or if at least ϵ-fraction of the points need to be relabeled or removed for the points to be linearly separable. Using result 1 above, we directly get a query complexity bound of $O(d/\epsilon)$. We also give a matching lower bound for this problem which implies a lower bound for result 1.
3. We obtain a result for property testing of many classical LP-Type problems. In particular, we consider the following problems:
   - *Smallest enclosing ball*: Accept if a set of points in $\mathbb{R}^d$ can be covered by a ball of radius r and reject if at least ϵ-fraction of the points need to be removed to be able to be covered by a ball of radius r. This problem has been previously studied in [1]. We improve upon the upper bound obtained in this paper in the case of two-sided error by getting a tight query complexity of $O(d/\epsilon)$ queries (also see point 4 below). Note that the algorithm in [1] has one sided error but has slightly worse query complexity.
   - *Smallest intersecting ball*: Accept if a set of closed convex bodies in $\mathbb{R}^d$ can all be intersected by a ball of radius r and reject if at least ϵ-fraction of the convex bodies need to be removed to be able to be intersected by a ball of radius r.
   - *Smallest volume annulus*: Accept if a set of points in $\mathbb{R}^d$ can be enclosed in an annulus of volume V and reject if at least ϵ-fraction of the points need to be removed to be encloseable by an annulus of volume V.

   In all these cases, it is known that the dimension of the LP-Type problem is linearly related to the dimension of the points in S, so we get an upper bound of $O(d/\epsilon)$ queries.
4. We get a matching lower bound of $\Omega(d/\epsilon)$ queries for the smallest enclosing ball problem and the smallest intersecting ball problem. This provides a lower bound for the radius cost of clustering considered by Alon et al. in [1] in the case of 1 cluster.

▶ Remark 6. Note that there are also many examples of LP-Type problems where the constraints in S describe points in dimension d but the dimension of the LP-Type problem is not a linear function of d. For example, if $\varphi(S)$ is the smallest ellipsoid that encloses the set of points in S which are in $\mathbb{R}^d$, then (S, φ) has dimension $O(d^2)$ as a LP-Type problem [12]. We did not explicitly highlight these problems but our approach also gives an upper bound on the query complexity for the property testing versions of these problems.

3 General Algorithm for Property Testing of LP-Type problems

We now present our general algorithm, LP-TYPE TESTER, for property testing of LP-Type problems as defined in Definition 4. Given a LP-Type problem (S, φ), Our algorithm first samples a subset R of $O(\delta/\epsilon)$ constraints from S where δ is the dimension of the LP-Type problem. It then calculates the value of φ on the sampled subset. After this step, an additional $O(1/\epsilon)$ constraints are sampled randomly from S. If $\varphi(R \cup \{x\})$ differs from $\varphi(R)$ where x is any of the additional random constraints, then our algorithm outputs reject. Otherwise, the algorithm outputs accept. We present our approach in Algorithm 1 along with our main theorem, Theorem 7 which proves the correctness of Algorithm 1.

Algorithm 1 LP-Type Tester.

```
Input  : $\delta, \epsilon, k$, query access to constraints in $S$
Output : accept or reject
1 $r \leftarrow \lceil 10\delta/\epsilon \rceil$
2 $R \leftarrow$ random sample of size $r$ of constraints from $S$.
3 if $\varphi(R) > k$ then
4     Output reject and abort.
5 for $2/\epsilon$ rounds do
6     $x \leftarrow$ uniformly random constraint of $S \setminus R$
7     if $\varphi(R \cup \{x\}) \neq \varphi(R)$ then
8         Output reject and abort.
9 Output accept.
```

▶ **Theorem 7** (Correctness of LP-Type Tester). *Given an LP-Type problem (S, φ) of dimension δ and parameters k and ϵ, the following statements hold with probability at least $2/3$:*

- ***Completeness case:*** *LP-Type Tester outputs accept $\varphi(S) \leq k$.*
- ***Soundness Case:*** *LP-Type Tester outputs reject if at least $\epsilon|S|$ constraints need to be removed from S for $\varphi(S) \leq k$ to hold.*

▶ Remark 8. Note that the query complexity of Algorithm 1 is $O(\delta/\epsilon)$ which is independent of $|S|$, the number of constraints.

3.1 Overview of the proof

To prove the correctness of LP-Type Tester, we analyze the completeness case and the soundness case separately. For the soundness case, we show that with sufficiently large probability, either $\varphi(R) > k$ or LP-Type Tester outputs reject during the second sampling phase where we sample an additional $O(1/\epsilon)$ constraints. To show this, we use the locality property of LP-Type problems (see Definition 1) to show that there must be "many" x such that $\varphi(R \cup \{x\}) \neq \varphi(R)$. To analyze the completeness case, we use the *Sampling Lemma*, Lemma 11, to show that there are "few" x such that $\varphi(R \cup \{x\}) \neq \varphi(R)$ so that Algorithm 1 outputs accept with sufficiently large probability.

Before we present the proof of Theorem 7, we present the Sampling Lemma as described above. This lemma has previously been used to study LP-Type problems. For completeness, we present a proof. For more information, see [26, 18, 11, 10]. Before we present the lemma, we introduce two new definitions.

▶ **Definition 9** (Violators and Extreme Elements). *For a subset $R \subseteq S$, define the violators and extreme elements of R as the following:*

- *Define the **violators** of R as the set $V(R) = \{s \in S \backslash R \mid \varphi(R \cup \{s\}) \neq \varphi(R)\}$.*
- *Define the **extreme elements** of R as the set $X(R) = \{s \in R \mid \varphi(R) \neq \varphi(R \backslash \{s\}\}$.*

▶ Remark 10. Note that s is a violator of R if and only if s is an extreme element of in $R \cup \{s\}$.

We now present the Sampling Lemma.

▶ **Lemma 11** (Sampling Lemma). *Let $v_r = \mathbb{E}[|V(R)|]$ and $x_r = \mathbb{E}[|X(R)|]$ where both expectations are taken over the random subsets R of S which have size r. Suppose $|S| = n$. Then for $0 \le r \le n$, we have*

$$\frac{v_r}{n-r} = \frac{x_{r+1}}{r+1}.$$

Proof. Let $\mathbf{1}\{\cdot\}$ denote an indicator variable. Note that

$$\begin{aligned}\binom{n}{r} v_r &= \sum_{R \in \binom{S}{r}} \sum_{s \in S \setminus R} \mathbf{1}\{s \text{ is a violator of } R\} = \sum_{R \in \binom{S}{r}} \sum_{s \in S \setminus R} \mathbf{1}\{s \text{ is extreme for } R \cup \{s\}\} \\ &= \sum_{Q \in \binom{S}{r+1}} \sum_{s \in Q} \mathbf{1}\{s \text{ is extreme for } Q\} = \binom{n}{r+1} x_{r+1}.\end{aligned}$$

The proof follows from the following calculation.

$$\frac{\binom{n}{r+1}}{\binom{n}{r}} = \frac{r!(n-r)!}{(r+1)!(n-r-1)!} = \frac{n-r}{r+1}.$$ ◀

▶ Remark 12. Note that (S, φ) does not need to be a LP-Type problem for the Sampling Lemma to hold true.

If (S, φ) is a LP-Type problem, there is a direct relationship between the expected number of violators and the dimension of (S, φ) as defined in 3. The following corollary also appears in many forms in literature (for instance [26, 18, 11, 4]) but we present its proof for completeness.

▶ **Corollary 13.** *Let (S, φ) be a LP-Type problem of dimension δ and let $|S| = n$. If $R \subseteq S$ is subset of size r chosen uniformly at random, then $v_r = \mathbb{E}[|V(R)|]$ satisfies*

$$v_r \le \frac{\delta(n-r)}{r+1}.$$

Proof. We show that for any set $R \subseteq S$, we have $|X(R)| \le \delta$. Then the corollary follows from Lemma 11. Let R' be the smallest subset of R such that $\varphi(R') = \varphi(R)$. We first claim that $V(R') = V(R)$. It is clear that $V(R) \subseteq V(R')$ by monotonicity (see Definition 1). For the other inclusion, consider $x \in V(R')$. If $x \notin V(R)$, we have $\varphi(R \cup \{x\}) = \varphi(R) = \varphi(R')$ so by locality, we have $\varphi(R') = \varphi(R' \cup \{x\})$ which contradicts the fact that $x \in V(R')$. Therefore, our claim holds true.

We now claim that R' is a basis as defined in Definition 2. Suppose for the sake of contradiction that R' is not a basis. Then there exists a $F \subset R'$ such that $\varphi(F) = \varphi(R')$. We now claim that $V(R') = V(F)$. It is clear that $V(R') \subseteq V(F)$. To show the other inclusion, let $x \in V(F)$. Then if x was not a violator of R', then $\varphi(R' \cup \{x\}) = \varphi(R') = \varphi(F)$ which would imply that $\varphi(F \cup \{x\}) = \varphi(F)$ by the locality property in Definition 1 which is false by definition. Hence, $V(F) = V(R') = V(R)$ which contradicts the minimality of R'. Therefore, R' is a basis.

Finally, we claim that if $x \in X(R)$ then $x \in R'$. This must be true because otherwise, we have $R' \subseteq R \setminus \{x\} \subseteq R$ which results in a contradiction by monotonicity. Finally, since $X(R) \subseteq R'$ and R' is a basis, it follows that $|X(R)| \le \delta$, as desired. ◀

▶ Remark 14. Corollary 13 holds for a larger class of problems than LP-Type problems called *violator spaces* ([11, 4]. However, we omitted this extra layer of abstraction since there are no additional natural property testing consequences from considering violator spaces over LP-Type problems.

Proof of Theorem 7. We first prove the soundness case. Consider the set R that was randomly sampled in step 2 of LP-TYPE TESTER. Assume that $\varphi(R) \leq k$ since this can only decrease the probability that our algorithm outputs reject. Now we claim that there must be at least $\epsilon|S|$ choices of x in step 6 of LP-TYPE TESTER that results in $\varphi(R \cup \{x\}) > \varphi(R)$ (so that we correctly output reject). To show this, note that if $\varphi(R) = \varphi(R \cup \{x\}) = \varphi(R \cup \{y\})$ for $x \neq y$ then by locality, it follows that $\varphi(R) = \varphi(R \cup \{x, y\})$. Therefore, if there are less than $\epsilon|S|$ choices of x in step 6 of LP-TYPE TESTER for some R such that $\varphi(R \cup \{x\}) > \varphi(R)$, then we would have $\varphi(R \cup R') = \varphi(R) \leq k$ where $|R \cup R'| \geq (1-\epsilon)|S|$ which would contradict our assumption that at least $\epsilon|S|$ constraints need to be removed from S for $\varphi(S) \leq k$ to hold true. Therefore, the probability our algorithm does not output reject in any of the $2/\epsilon$ rounds is at most

$$(1-\epsilon)^{2/\epsilon} \leq e^{-2} < \frac{1}{3} \tag{1}$$

which means that we output reject with probability at least 2/3, as desired.

We now prove Theorem 7 for the completeness case. Let $v_r = \mathbb{E}[|V(R)|]$. Since $r = |R| = 10\delta/\epsilon$, Corollary 13 gives us

$$v_r \leq \frac{\delta(|S| - r)}{r+1} \leq \frac{\epsilon|S|}{10}.$$

Therefore in the completeness case, the probability that a randomly chosen x satisfies $\varphi(R \cup \{x\}) \neq \varphi(R)$ is at most $\epsilon/10$. Since we choose $2/\epsilon$ random constraints, the probability we don't find such a x is at least

$$(1-\epsilon/10)^{2/\epsilon} \geq 1 - \frac{2}{10} > \frac{2}{3}. \tag{2}$$

Therefore, LP-TYPE TESTER outputs accept with probability at least 2/3, as desired. ◀

4 Property Testing Applications of LP-Type Tester

We now give applications of the framework we build in Section 3. We first consider the problem of testing feasibility of a set of linear inequalities. As a direct consequence, we can test if a set of labeled points can be linearly sepearable (either by linear hyperplanes or by functions that have a finite basis). These two applications will not be an immediate corollary of Theorem 7 since there is no objective function that we want to optimize, but our results follow from Theorem 7 with some slight modificatons.

We then consider direct applications of Algorithm 1 to some cannonical LP-Type problems such as the smallest enclosing ball. Theorem 7 gives direct upper bounds for property testing for these problems.

4.1 Testing Feasibility of a System of Linear Equations

We first begin by considering testing feasibility of a set of linear inequalities. Recall that in this problem, we have n linear constraints in $\mathbb{R}^d$ (such as $x_1 + \cdots + x_d \leq 1$) and we want to distinguish the following two cases with probability at least 2/3:

- The system of linear inequalities can all be mutually satisfied, i.e., the system is feasible (Completeness Case)
- At least $\epsilon|S|$ many of the constraints need to be removed (or flipped) for the system to be feasible (Soundness Case).

This is not exactly a LP-Type problem since we do not have an optimization function φ. We note that if φ was an indicator function for a subset of constraints being feasible then φ would break the locality condition in Definition 1. One way to see this is consider the case when A is the single constraint that $y \geq 0$, B is the set of two constraints $0 \leq y$ and $y \leq 1$, and x is the additional constraint that $y \geq 2$. We can see that A, B, and $A \cup \{x\}$ are all feasible, but $B \cup \{x\}$ is not, contradicting locality. However, we perform a slight modification of Algorithm 1 to create a new algorithm for this problem.

Our algorithm for this testing problem, LINEAR FEASIBILITY TESTER, uses the fact that if we pick any arbitrary $x \in \mathbb{R}^d$, then x will violate "many" of the linear constraints in S in the completeness case. In the soundness case, we use ideas from LP-TYPE TESTER and show that if we introduce an arbitrary linear optimization function (thus turning our problem into an instance of linear programming), then a solution that optimizes a small subset of the constraints will not violate "too many" of the other constraints. We present our algorithm below along with Theorem 15 that proves its correctness.

Algorithm 2 LINEAR FEASIBILITY TESTER.

```
Input  : $d, \epsilon$, query access to constraints of $S$
Output : Accept or Reject
1  $r \leftarrow \lceil 10d/\epsilon \rceil$
2  $R \leftarrow$ random sample of size $r$ of constraints from $S$
3  Create the linear program $L$: max $x_1$ subject to the constraints in $R$
4  $x \leftarrow$ solution of $L$
5  if $L$ is not feasible then
6      Reject and abort
7  for $2/\epsilon$ rounds do
8      $y \leftarrow$ uniformly random constraint of $S$
9      if $x$ does not satisfy $y$ then
10         Output reject and abort.
11 Output accept.
```

▶ **Theorem 15** (Correctness of LINEAR FEASIBILITY TESTER). *Given a set S of linear inequalities in $\mathbb{R}^d$, the following statements hold with probability at least $2/3$:*

- ***Completeness case:*** *LINEAR FEASIBILITY TESTER outputs accept if there exists $x \in \mathbb{R}^d$ that satisfies all of the constraints in S.*
- ***Soundness Case:*** *LINEAR FEASIBILITY TESTER outputs reject if at least $\epsilon|S|$ constraints need to be removed from S for S to be feasible.*

▶ Remark 16. Note that the query complexity of Algorithm 1 is $O(d/\epsilon)$ which is independent of $|S|$, the number of constraints. Furthermore, the runtime is polynomial in d/ϵ since we are solving a linear programs in d variables and $O(d/\epsilon)$ constraints.

Proof. The proof of the soundness case follows similarly to Theorem 7 using the fact that for any x, there are at least $\epsilon|S|$ constraints in x such that x violates these constraints. Then the probability that LINEAR FEASIBILITY TESTER outputs reject in this case can be calculated to be at least $2/3$ using the same bound as Eq. (1) in the proof of Theorem 7.

For the completeness case, we note that if we introduce the optimization function $\varphi(S) = \max x_1$ subject to the constraints in S, then (S, φ) is an LP-Type problem of dimension d (assuming that the constraints are non degenerate which can be assumed by perturbing the constraints and then taking the limit of the perturbation to 0. For more details, see [6, 25]). Now let x be the solution to the linear program that we solved in Step 4 of Linear Feasibility Tester. Using Corollary 13, we know that if $|R| = 10\lceil d/\epsilon \rceil$, then the number of constraints v_r in S that satisfy $\varphi(R \cup \{y\}) \neq \varphi(R)$ is at most $v_r \leq (d|S|)/(10d/\epsilon) = \epsilon|S|/10$ in expectation. Knowing that x not satisfying y implies that y is a violator of R, the probability that x does not satisfy a randomly chosen y is at most $\epsilon/10$. Thus, using the exact calculation as in Eq.(2) of Theorem 7, we have that Linear Feasibility Tester outputs accept in the completeness case with probability at least 2/3, as desired. ◀

In Section 6 we give a *tolerant* tester for testing linear feasibility. A tolerant tester accepts instances that are ϵ-close and rejects instances that are $c\epsilon$-far for some constant $c > 1$.

4.2 Testing if Labeled Points can be Linearly Separated

As a direct consequence of the Theorem 15, we can test if a set of points in d dimensions labeled $\{+1, -1\}$ can be linearly separated. More formally, we have the following corollary.

▶ **Corollary 17.** *Given a set S of points in $\mathbb{R}^d$ with labels in $\{+1, -1\}$, the following statements hold with probability at least $2/3$:*

- ***Completeness case:*** *Linear Feasibility Tester outputs accept if there exists a hyperplane that separates the two sets of labeled points.*
- ***Soundness Case:*** *Linear Feasibility Tester outputs reject if at least $\epsilon|S|$ points need to be removed (or relabeled) for S to be linearly sepearable.*

Proof. The proof follows directly from the fact that we can write a linear inequality that represents a separating hyperplane. For example, if $p \in S$ is labeled 1, we want to find x such that $p^T x \geq 1$ and if p is labeled -1, we want to find x such that $p^T x \leq -1$. ◀

We consider generalizations of this problem where we wish to separate labelled points by arbitrary functions, rather than just linear hyperplanes. In Section 7 we address the issue of separating using arbitrary functions when we know the basis of the functions, and the case of multiple labels.

4.3 Upper Bounds for Canonical LP-Type Problems

We now give direct applications of LP-Type Tester to some canonical LP-Type problems. The correctness of these applications follows directly from Theorem 7. Our list is not exhaustive and we only consider some of the more well known LP-Type problems. In all of the following problems, Theorem 7 tells us that the following statements hold with probability at least 2/3:

- LP-Type Tester outputs accept if $\varphi(S) \leq k$ (Completeness Case)
- LP-Type Tester outputs reject if at least $\epsilon|S|$ constraints need to be removed from S for $\varphi(S) \leq k$ to hold (Soundness Case).

Our results are the following:

- *Smallest enclosing ball*: In this problem, $\varphi(S)$ is the radius of the smallest enclosing ball of a set of points S in $\mathbb{R}^d$. It is known that the dimension of this LP-Type problem is $d + 1$ (see [12]) so we can test if $\varphi(S) \leq k$ with query complexity $O(d/\epsilon)$ queries.

- *Smallest intersecting ball*: In this problem, $\varphi(S)$ is the smallest radius ball that intersects a set of closed convex bodies S in $\mathbb{R}^d$. The dimension of this LP-Type problem is $O(d)$ ([12]) so we can test if $\varphi(S) \leq k$ with query complexity $O(d/\epsilon)$ queries.
- *Smallest volume annulus*: In this problem, $\varphi(S)$ is the volume of the smallest annulus that contains a set of points S in $\mathbb{R}^d$. Again, the dimension of this LP-Type problem is $O(d)$ ([12]) so we can test if $\varphi(S) \leq k$ with query complexity $O(d/\epsilon)$.

5 Lower Bounds

In this section, we give matching lower bounds for all the testing problems that we considered in Section 4.

5.1 Lower Bound for Testing Feasibility of Linear Constraints

Since linear separability is a special case of feasibility of linear constraints, we can lower bound the necessary query complexity of the latter by providing one for the former. In particular, we aim to show that $\Omega(d/\epsilon)$ queries are needed to determine if a set of points in d dimensions is linearly separable. By the reduction of linear separability to feasibility of linear constraints, this implies that $\Omega(d/\epsilon)$ constraint queries are needed to test feasibility of a system of linear constraints, which matches our upper bound.

Our overall approach is to first introduce a set of $O(d)$ points in $\mathbb{R}^d$ that have the property that if we do not look at a large enough collection of these points, they can be separated by a hyperplane even with arbitrary labels. However, there will exist a labeling of all of the points such that "many" of the points will have to be removed or relabeled for this labeling to be separated. The existence of these points is given in Lemma 18 (and is inspired by the moment curve).

Then, repeating these points with carefully chosen multiplicities allows us to construct our set S of points. Then a coupon collector argument gives us our desired lower bound on the query complexity. This argument is formalized in the proof of Theorem 19.

▶ **Lemma 18.** *There exists a set S of $3d+1$ points in $\mathbb{R}^d$ that satisfy the following conditions:*

1. *There exists a labeling of the points of S such that at least d points have to be relabeled for the points to be linearly separable.*
2. *Any subset of points of S of size $d+1$ with arbitrary labels in $\{-1, 1\}$ is linearly separable.*

Proof. We construct our set S as follows. Let x_i be the point $(i^1, \cdots, i^d) \in \mathbb{R}^d$ for $1 \leq i \leq 3d+1$ (note that this set of points is referred to as the moment curve). We prove the first claim using a standard relationship between the moment curve and polynomials. Assign the point x_i to the label $(-1)^i$. Let k be the number of relabeled points such that S is linearly separable. Then there exists $w \in \mathbb{R}^d$ and $w_0 \in \mathbb{R}$ such that $\mathrm{Sign}(x_i^T w + w_0)$ matches the label of every point $x_i \in S$. In other words, there exists a polynomial $P(x) = \sum_{j=0}^d c_j x^j$ such that $\mathrm{Sign}(P(i))$ matches the label of x_i. Now note that if there are two consecutive indices i and $i+1$ that have different labels, then P must have a root in the interval $(i, i+1)$. Originally, there are $3d$ such alternating intervals. Now note that the relabeling of any point can decrease the total number of such alternating intervals by at most 2. Hence after k relabelings, there must be at least $3d - 2k$ alternating intervals. However, since P is a d degree polynomial, it must have at most d roots which means $3d - 2k \leq d$ and therefore, $k \geq d$, as desired.

We now prove the second claim. Let $x_{a_1}, \cdots, x_{a_{d+1}}$ be a subset of $d+1$ points of S. Without loss of generality, suppose that $a_1 < \cdots < a_{d+1}$. We now show that for every labelings of these $d+1$ points, there exists a polynomial of degree d such that the sign of $P(a_i)$ matches the label of x_{a_i}. Towards this goal, pick t elements $b_1, \cdots, b_t$ of the set $\{a_2, \cdots, a_{d+1}\}$ where $t \leq d$. Consider the $t+1$ intervals

$$[a_1, b_1), [b_1, b_2), \cdots, [b_{t-1}, b_t), [b_t, a_{d+1}+1).$$

We can then find a polynomial of degree d such that

- the sign of P is constant on $I \cap \{a_1, a_2, \cdots, a_{d+1}\}$ where I is any of the $t+1$ intervals above,
- the sign of P alternates between consecutive intervals.

This is possible since we are only specifying the value of P on $d+1$ locations. Now the total number of labelings described by all possible choices of P is given by $2\sum_{t=0}^{d}\binom{d}{t} = 2^{d+1}$ where the factor of 2 comes from specifying the sign of P on the first interval. Note that 2^{d+1} is exactly the total number of different ways to label $d+1$ points, which proves the second claim. ◀

With Lemma 18 on hand, we can prove our desired lower bound on the query complexity.

▶ **Theorem 19.** *Any algorithm that tests if a set S of labeled points in d dimensions can be linearly separated requires $\Omega(d/\epsilon)$ queries.*

Proof. Let $|S| = n$. We create two families of n points in $\mathbb{R}^d$ with a specific labeling such that any S from one family can be linearly separated while any S from the other family is ϵ-far from being linearly separable. First, consider the set of $3d+1$ points supplied by Lemma 18 and the labeling from part 1 of the lemma. The first family $\mathcal{F}_1$ consists of picking a subset of $d+1$ of these points (with the labeling above), repeating d of these points $n\epsilon/d$ times, and repeating the remaining point $(1-\epsilon)n$ times. The second family $\mathcal{F}_2$ (again with the same labeling) consists of picking all of the $3d+1$ points from Lemma 18, repeating some $3d$ of these points with multiplicity $n\epsilon/(3d)$, and repeating the last point with multiplicity $(1-\epsilon)n$.

By Lemma 18, we know that if S is from $\mathcal{F}_1$ then S is linearly separable while if S is from $\mathcal{F}_2$, then S is at least $\epsilon/(3d) \cdot d = O(\epsilon)$-far from separable. Any algorithm that queries points randomly must discover at least $d+1$ *unique* points out of the points that were repeated $n\epsilon/d$ time from any S in $\mathcal{F}_2$ to discover that this S is $O(\epsilon)$-far from separable (otherwise, the points look separable). Call points that are identical “groups”. Now given a random point from S, the probability of hitting any one group is $\epsilon/(3d)$. Therefore by coupon collector, the expected number of queries required to hit at least $d+1$ of these $3d$ groups is at least

$$\frac{1}{\epsilon}\left(\frac{3d}{3d} + \frac{3d}{3d-1} + \cdots + \frac{3d}{3d-d}\right) = \frac{3d}{\epsilon}(H_{3d} - H_{2d-1}) = \Theta\left(\frac{d}{\epsilon}\right). \qquad ◀$$

As a corollary, we have the following lower bound as well. This is due to the reduction from linear separability to linear program feasibility from the proof of Corollary 17.

▶ **Theorem 20.** *Any algorithm that tests if n linear inequalities in d dimensions are feasible requires $\Omega(d/\epsilon)$ queries.*

We now give matching query complexity lower bounds for the LP-Type problems that we considered in Section 4.

5.2 Lower bound for Testing Smallest Enclosing Ball

We first give a lower bound for property testing the radius of the smallest enclosing ball of a set of points. Our approach is to first construct a set of points in $\mathbb{R}^j$, for any j, whose smallest enclosing ball can be calculated exactly. This set of points will have the property that a small enough subset of the points will have a significantly smaller enclosing ball. Therefore, if an algorithm does not query enough points, it will incorrectly believe that this set of points can be covered by a ball of small radius. Our construction for this case will be a regular simplex and explained below. First we prove an auxiliary lemma.

▶ **Lemma 21.** *The radius of the circumcircle of a unit simplex in $\mathbb{R}^j$ is $\sqrt{j}/(\sqrt{2(j+1)})$.*

Proof. Note that we can embed a regular j-simplex in $\mathbb{R}^{j+1}$ using the coordinates $\{e_i\}_{i=1}^{j+1}$ where e_i is the all zero vector with a single 1 in the ith coordinate. This simplex has edge length $\sqrt{2}$ so we can scale appropriately to find the circumcircle of a unit simplex. Now the centroid of this simplex is easily seen to be located at $(1/(j+1), \cdots, 1/(j+1))$ which means that the circumcircle has radius

$$\sqrt{\left(1 - \frac{1}{j+1}\right)^2 + \frac{j}{(j+1)^2}} = \sqrt{\frac{j}{j+1}}.$$

Now scaling by $1/\sqrt{2}$ gives us the desired value. ◀

▶ **Theorem 22.** *Any algorithm that tests if a set of n points in $\mathbb{R}^d$ can be enclosed by a ball of radius k, where k is given, requires $\Omega(d/\epsilon)$ queries.*

Proof. Let k be fixed. We construct two families of points in $\mathbb{R}^{O(d)}$ such that any S from one family can be enclosed by a ball of radius k while any S from the second family is ϵ-far from being enclosed by a ball of radius k. Before constructing these families, we first pick ℓ such that the regular simplex of side length ℓ in $\mathbb{R}^{d+1}$ has circumradius k.

Now to create the first family $\mathcal{F}_1$, we first pick any $d+1$ points of the regular simplex with side length ℓ in $\mathbb{R}^{3d+1}$. Then we repeat one of these points with multiplicity $(1-\epsilon)n$ and we repeat the other d points with multiplicity $n\epsilon/d$ each. To create the second family $\mathcal{F}_2$, we pick a point of the regular simplex with side length ℓ in $\mathbb{R}^{3d+1}$, repeat it with multiplicity $(1-\epsilon)n$, and repeat the other $3d$ points with multiplicity $n\epsilon/(3d)$. Finally, let S be a set of n points from $\mathcal{F}_2$. From Lemma 21, we can check that the circumradius of a regular unit simplex is an increasing function of the dimension and that any subset of the vertices of a regular simplex is a regular simplex itself. Therefore, the smallest radius of the points in S is much larger than k and S is $O(\epsilon)$-far from being encloseable by a ball of radius k. However, similar to the argument in Theorem 19, any algorithm that rejects S must have discovered at least $d+1$ distinct "groups" of repeated points. By the same coupon collector argument as in the proof of Theorem 19, we have that this task takes at least $\Omega(d/\epsilon)$ queries in expectation. ◀

As a simple application of Theorem 22, we get the following lower bounds as well.

▶ **Corollary 23.** *Any algorithm for testing the smallest intersecting ball for n convex bodies in $\mathbb{R}^d$ requires $\Omega(d/\epsilon)$ queries.*

Proof. The proof follows from the fact that a set of singleton points is also a set of convex bodies. In this case, the smallest intersecting ball is equivalent to the smallest ball that encloses these points. Therefore, the same lower bound as in Theorem 22 holds. ◀

6 Tolerant Tester for Testing Feasibility of Linear Constraints

We generalize our argument in Section 4.1 by giving a *tolerant* tester for testing feasibility of a system of linear constraints. In the tolerant version, we output accept if there only “few” constraints need to be removed for a set of linear inequalities to be feasible. More formally, we wish to distinguish the following two cases with probability at least 2/3:

- At most $c\epsilon|S|$ many inequalities in S need to be removed for S (or flipped) for S to be feasible, i.e., S is $c\epsilon$-close to being feasible for some fixed positive $c < 1$ (Completeness Case).
- At least $\epsilon|S|$ many of the constraints need to be removed (or flipped) for the system to be feasible (Soundness Case).

Our approach is a slightly modified version of Linear Feasibility Tester, Algorithm 2, that we presented in Section 4.1. The challenge here is the completeness case where we must accept if we only have a “few” bad constraints. To accomplish this, we carefully select a solution to a small linear program that we run. For more details, see Algorithm 3. Our main theorem in this section, Theorem 24 shows that we can perform *tolerant* testing using the same query complexity we used for the non tolerant tester in Section 4.1, namely $O(d/\epsilon)$. However, as we will explain below, the *running time* of Algorithm 3, Tolerant Linear Feasibility Tester, is exponential in the running time of Algorithm 2. Our algorithm, Tolerant Linear Feasibility Tester, is presented below.

Algorithm 3 Tolerant Linear Feasibility Tester.

```
Input  : $d, \epsilon$, query access to constraints of LP
Output : Accept or Reject
1  $r \leftarrow \lceil 10d/\epsilon \rceil$
2  $R \leftarrow$ random sample of size $r$ of constraints from $S$
3  $x \leftarrow$ solution of the largest subset $R'$ of $R$ such that the linear program $L$: $\max x_1$
     subject to the constraints in $R'$ is feasible
4  if No $L$ is not feasible then
5      Reject and abort
6  for $2/\epsilon$ rounds do
7      $y \leftarrow$ uniformly random constraint of $S$
8      if $x$ does not satisfy $y$ then
9          Output reject and abort.
10 Output accept.
```

Unlike Linear Feasibility Tester where we run a linear program, we solve a slightly different program given in step 3 of Tolerant Linear Feasibility Tester. The step determines the largest feasible subset of these constraints. Note that this step is clearly exponential in the number of constraints (which is $O(d/\epsilon)$). Therefore, the overall *runtime* of Tolerant Linear Feasibility Tester will be exponential in the runtime of Linear Feasibility Tester. The correctness of Tolerant Linear Feasibility Tester is proven in Theorem 24.

▶ **Theorem 24** (Correctness of Tolerant Linear Feasibility Tester). *Given a set S of linear inequalities in $\mathbb{R}^d$, there exists a constant $c < 1$ such that the following statements hold with probability at least $2/3$:*

- ***Completeness case:*** *Tolerant Linear Feasibility Tester outputs accept if there exists $x \in \mathbb{R}^d$ that satisfies $(1 - c\epsilon)|S|$ of the constraints in S.*
- ***Soundness Case:*** *Tolerant Linear Feasibility Tester outputs reject if at least $\epsilon|S|$ constraints need to be removed from S for S to be feasible.*

▶ Remark 25. Note that the query complexity of Algorithm 3 is $O(d/\epsilon)$ which is independent of $|S|$, the number of constraints.

Proof. Note that the proof of the soundness case is identical to the proof of the soundness case in Theorem 15 since for any x we find in step 3 of Tolerant Linear Feasibility Tester, there exists at least $\epsilon|S|$ choices of y in step 7 such that x does not satisfy the constraint y. Then a similar calculation as in Eq. (1) implies that we reject with probability at least $2/3$.

We now focus on the completeness case where we know there is a subset of $(1 - c\epsilon)|S|$ constraints that are feasible. We call this the *good* set, and the rest, the *bad* set. Consider the sample R from step 2 of Tolerant Linear Feasibility Tester. The expected number of constraints from the good set in R is $(1 - c\epsilon)r$. This means at most $c\epsilon r$ constraints in R come from the bad set in expectation. Hence with probability at least $9/10$, we know that the number of constraints from the bad set is at most $10c\epsilon r$ by Markov's inequality, which means the number of constraints coming from the good set is at least $(1 - 10c\epsilon)r$. We condition on this event. Now note that one valid subset R' to use in step 3 of Tolerant Linear Feasibility Tester is to take all the constraints coming from the good set only. This results in $|R'| \ge (1 - 10c\epsilon)$. Since we are maximizing $|R'|$, this means that at most $10c\epsilon r$ of the constraints coming from the good set that are in R will not be included in R'. Thus, x satisfies at least $(1-20c\epsilon)r$ constraints in the good set with probability at least $9/10$. Now we proceed similarly as the proof of Theorem 15. By Corollary 13, the probability that x violates any other constraint in the good set is at most

$$\frac{d(n' - r + 1)}{n'(r - d)} \le \frac{dn'}{10dn'/\epsilon} = \frac{\epsilon}{10}$$

where n' is the size of the good set. Furthermore, x can possibly violate any constraint in the bad set which means that the probability x violates any other constraint is at most $\epsilon/10 + c\epsilon < \epsilon/6$ for sufficiently small c, i.e. $c < 1/15$. Then, the probability that we find such a constraint in $2/\epsilon$ rounds is at most

$$1 - \left(1 - \frac{\epsilon}{6}\right)^{2/\epsilon} \le 1 - \left(1 - \frac{1}{3}\right) = \frac{1}{3}.$$

Therefore, we accept with probability at least $2/3$, as desired. Note that we can take any $c < 1/15$ in the statement of the Theorem for instance. ◀

7 Separating Points with Arbitrary Functions and Multiple Labels

7.1 Separating labeled points using arbitrary functions

We can generalize our result from Section 4.2 by separating labeled points using arbitrary *functions*: given a family of functions $\mathcal{F}$, we can ask if there is a $f \in \mathcal{F}$ such that $f(p) > 0$ for all points with a particular label and $f(p) < 0$ for all the points with the other label.

We now translate this problem to a setting with linear inequalities. Our approach is standard in machine learning and is known as feature maps. If the family $\mathcal{F}$ has a finite basis $f_1, \cdots, f_k$, meaning that every $f \in \mathcal{F}$ is a linear combination of $f_1, \cdots, f_k$, then we can create a system of linear inequalities as follows. For each point $p \in S$, we can make a new constraint which is $(f_1(p), \cdots, f_k(p))x \geq 1$ (note there that x is a column vector of variables) if p has one particular label or ≤ -1 if p has another label. Then this system of linear constraints is feasible iff there are scalars $a_1, \cdots, a_k$ such that $\sum_i a_i f_i(p) \geq 0$ for all p with one label and $\sum_i a_i f_i(p) \leq 0$ for all p with the other label. Then our separating function is precisely $f = \sum_i a_i f_i$. Note that in this formulation, we have k variables. Thus, the query complexity is $O(k/\epsilon)$.

As an example, we consider the case that $\mathcal{F}$ is the family of polynomials in d variables with degree $\leq t$. The basis of this family is all the possible terms of the form $x_1^{t_1} \cdots x_d^{t_d}$ where the t_i are non-negative and add to at most t. By a standard balls and bins argument, the number of these terms is $\binom{t+d}{d}$. For constant t, this is $O(d^t)$, which means that our system of linear constraints has $O(d^t)$ variables. Thus, the query complexity is $O(d^t/\epsilon)$.

7.2 Separating Points with Multiple Labels

Suppose that in Section 4.2, instead of assigning each point one of 2 labels, we instead chose to assign it one of $\ell \geq 2$ labels. One common interpretation of separability for this setup is to check if each of the $\binom{\ell}{2}$ pairs of label sets are separable. We modify our notion of ϵ-far to reflect this.

▶ **Definition 26.** *S is ϵ-far from linearly separable if at least $\epsilon|S|$ many labels in S have to be changed for S to be separable.*

If such a data set is ϵ-far from separable, then some subset with consisting of two labels must be $\epsilon/\binom{\ell}{2}$-far from separable. As such, we can consider an algorithm that runs Algorithm Linear Feasibility Tester on each pair of labels with $\epsilon' = \epsilon/\binom{\ell}{2}$ and outputs accept if all these tests output accept. We need to reduce the error probability for each pair such that the overall error probability of outputting the incorrect answer (acquired by a union bound) is still at most 1/3. This can be done by using a stronger version of the original algorithm where we run it $O(\log \ell)$ times and taking the majority answer. By a standard Chernoff bound argument, the probability this process gives the wrong answer is at most say $1/\ell^3$. Thus, we can distinguish separability in this case by running this stronger version over all pairs of distinct labels, resulting in $O(\ell^2 \log \ell)$ instances of Linear Feasibility Tester, using $\epsilon' = \epsilon/\binom{\ell}{2}$. So, the total query complexity will be $O(d\ell^4 \log \ell/\epsilon)$.

Additionally, the completeness case has error at most $\binom{\ell}{2}1/\ell^3 = o(1)$ by a Union Bound argument. Clearly, the soundness case has error at most $1/\ell^3$, since there is one pair of distinct labels which is ϵ'-far from separable.

References

1 N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 240–250, November 2000. doi:10.1109/SFCS.2000.892111.

2 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. The power and limitations of uniform samples in testing properties of figures. *Algorithmica*, 81(3):1247–1266, March 2019. doi:10.1007/s00453-018-0467-9.

3 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing convexity of figures under the uniform distribution. *Random Structures & Algorithms*, 54(3):413–443, 2019. doi:10.1002/rsa.20797.

4 Yves Brise and Bernd Gärtner. Clarksons algorithm for violator spaces. *CoRR*, abs/0906.4706, 2009. arXiv:0906.4706.

5 H. Chen, M. Valeriote, and Y. Yoshida. Testing assignments to constraint satisfaction problems. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 525–534, October 2016. doi:10.1109/FOCS.2016.63.

6 Kenneth L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, March 1995. doi:10.1145/201019.201036.

7 Artur Czumaj and Christian Sohler. Property testing with geometric queries. In Friedhelm Meyer auf der Heide, editor, *Algorithms — ESA 2001*, pages 266–277, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

8 Artur Czumaj and Christian Sohler. Abstract combinatorial programs and efficient property testers. *SIAM J. Comput.*, 34(3):580–615, March 2005. doi:10.1137/S009753970444199X.

9 Artur Czumaj, Christian Sohler, and Martin Ziegler. Property testing in computational geometry. In Mike S. Paterson, editor, *Algorithms - ESA 2000*, pages 155–166, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

10 B. Gärtner and E. Welzl. A simple sampling lemma: Analysis and applications in geometric optimization. *Discrete & Computational Geometry*, 25(4):569–590, April 2001. doi:10.1007/s00454-001-0006-2.

11 Bernd Gärtner, Jiří Matoušek, Leo Rüst, and Petr Škovroň. Violator spaces: Structure and algorithms. In Yossi Azar and Thomas Erlebach, editors, *Algorithms – ESA 2006*, pages 387–398, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

12 Bernd Gärtner and Emo Welzl. Linear programming — randomization and abstract frameworks. In Claude Puech and Rüdiger Reischuk, editors, *STACS 96*, pages 667–687, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

13 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.

14 Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, July 1998. doi:10.1145/285055.285060.

15 Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 289–298, New York, NY, USA, 1998. ACM. doi:10.1145/276698.276767.

16 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/s00453-001-0078-7.

17 Steve Hanneke. The optimal sample complexity of pac learning. *J. Mach. Learn. Res.*, 17(1):1319–1333, January 2016.

18 J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4):498–516, October 1996. doi:10.1007/BF01940877.

19 Krzysztof Onak. Testing properties of sets of points in metric spaces. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 515–526, 2008. doi:10.1007/978-3-540-70575-8_42.

20 Luis Rademacher and Santosh Vempala. Testing geometric convexity. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science*, pages 469–480, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

21 Sofya Raskhodnikova. Approximate testing of visual properties. In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, pages 370–381, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

22 Sofya Raskhodnikova. Approximate testing of visual properties. In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, pages 370–381, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

23 Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and TrendsÂ® in Theoretical Computer Science*, 5(2):73–205, 2010. doi:10.1561/0400000029.

24 Dana Ron and Gilad Tsur. Testing properties of sparse images. *ACM Trans. Algorithms*, 10(4), August 2014. doi:10.1145/2635806.

25 Raimund Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, 6(3):423–434, September 1991. doi:10.1007/BF02574699.

26 Micha Sharir and Emo Welzl. A combinatorial bound for linear programming and related problems. In Alain Finkel and Matthias Jantzen, editors, *STACS 92*, pages 567–579, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.