Testing Geometric Properties [*] [†]

Artur Czumaj [‡] **Christian Sohler** [§] **Martin Ziegler** [¶]

Abstract

In this paper we study property testing for basic geometric properties. We aim at developing efficient algorithms which determine whether a given (geometrical) object has a predetermined property Q or is "far" from any object having the property. We show that many basic geometric properties have very efficient testing algorithms, whose running time is significantly smaller than the object description size. We present optimal property testing algorithms for the problem of testing disjointness of geometric objects, disjointness of V-polytopes, convex position property of points sets, and of testing Delaunay triangulations.

[*]The results presented in this paper appeared in a preliminary form *as a part* of the paper "Property testing in computational geometry," in *Proceedings of the 8th Annual European Symposium on Algorithms (ESA'00)*, pages 155–166, volume 1879 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2000 [9].

[†]Research supported in part by NSF grants ITR-CCR-0313219, CCR-0105701, and by DFG grant Me 872/8-2.

[‡]Department of Computer Science, University of Warwick, Coventry, CV4 7AL, United Kingdom, czumaj@dcs.warwick.ac.uk.

[§]Heinz Nixdorf Institute and Department of Computer Science, University of Paderborn, D-33102 Paderborn, Germany, csohler@upb.de.

[¶]Heinz Nixdorf Institute and Department of Computer Science, University of Paderborn, D-33102 Paderborn, Germany, ziegler@uni-paderborn.de.

1 Introduction

Property Testing is the computational task to decide whether a given object (e.g., a graph, a function, or a point set) has a certain predetermined property (e.g., bipartiteness, monotonicity, or convex position) or is *"far"* from every object having this property. If the input object neither has the property nor is far from it, the algorithm may answer arbitrarily. The concept of property testing was first formulated by Rubinfeld and Sudan in the context of program checking [23]. Property testing arises naturally in the context of program verification, learning theory, and, in a more theoretical setting, in probabilistically checkable proofs. For example, in the context of program checking, one may first choose to test whether the program's output satisfies certain properties before checking that it is as desired. This approach is a very common practice in software development, where it is (typically) infeasible to require to formally test that a program is correct, but by verifying whether the output satisfies certain properties one can gain a reasonable confidence about the quality of the program's output. The study of property testing for *combinatorial objects* with focus on graphs was initiated by Goldreich et al. [15]. In many follow-up papers the concept of property testing has been applied to different classes of objects including graphs, matrices, languages, etc. For recent surveys of results on property testing we refer the reader to [12, 14, 22].

In this paper we consider property testing as applied to *geometric object in Euclidean spaces*. We investigate property testing for some basic and most representative problems/properties in computational geometry. The main goal of this research is to develop algorithms which perform only a very small (sublinear, polylogarithmic, or even a constant) number of operations in order to check with a reasonable confidence the required property. We study the following family of tasks:

> *Given oracle access to an unknown geometric object, determine whether the object has a certain predefined property or is "far" from any object having this property.*

Distance between objects is measured in terms of a *relative edit distance*. We say an object X (e.g., a set of points in $\mathbb{R}^d$) from a class $\mathcal{C}$ (e.g., a class of all point sets in $\mathbb{R}^d$) is ϵ*-far from satisfying a property* Q (over $\mathcal{C}$) if no object Y from $\mathcal{C}$ which differs from X in no more than an ϵ-fraction of places (e.g., Y can be constructed from X by adding and removing no more than $\epsilon \cdot |X|$ points) satisfies Q.

The notion of "*oracle access*" corresponds to the representation of the objects in $\mathcal{C}$ and is problem dependent. For example, if a geometric object is defined as a collection of points in Euclidean space, then it is reasonable to require that the oracle allows the algorithm to ask only on the coordinate-position of each single input point. If however, a geometric object is, say, a polygon, then the oracle shall typically allow to query for the position of each point as well as for each single neighbor of each point in the polygon. In this paper we shall always use the most natural notions of "oracle" for each problem at hand.

An ϵ*-tester* for a property Q is a (typically randomized) algorithm that always accepts[1] any object satisfying Q and with probability at least $\frac{2}{3}$ rejects any object being ϵ-far from satisfying Q [2]. There are two types of possible complexity measures of a tester we focus on: the *query* complexity and the *running time* complexity of an ϵ-tester. The query complexity of a tester is measured by the number of queries to the oracle, while the running time complexity counts also the time needed by the algorithm to perform other tasks (e.g., to compute a convex hull of some point set).

[1] In this paper we assume a *one-sided error* model, though in the literature also a *two-sided error* model has been considered. In the two-sided error model one aims at distinguishing with probability at least $\frac{2}{3}$ between the case an object X satisfies Q and the case of X being ϵ-far from satisfying Q.

[2] One could consider a confidence parameter δ rather than fixing an error bound $\frac{1}{3}$. In that case one could apply here standard amplification techniques and thus we have decided to fix the error bound.

To exemplify the notion of property testing, let us consider the standard geometrical property Q of a point set P in Euclidean space $\mathbb{R}^d$ of being in *convex position* (a point set is in *convex position* if every point of the set is a vertex of its convex hull). In this case, we aim at designing an algorithm that for a given oracle access to P has to decide whether P is in convex position or P is "ϵ-far" from convex position. Here, we say a set P is *ϵ-far from convex position*, $0 \le \epsilon \le 1$, if for every subset $X \subseteq P$ of size at most $\epsilon \cdot |P|$, the set $P \setminus X$ is not in convex position. To design testers we assume that the oracle allows to query for the size of P and for the position of each ith point in P, $1 \le i \le n$.

In this paper we design ϵ-testers for some basic properties in computational geometry:

disjointness of geometric objects: property that an arrangement of objects is *intersection-free*,

disjointness of polytopes: property that two polytopes are *intersection-free*,

convex position: property that a set of points is in *convex position*, and

Delaunay triangulation: property that a triangulation is a *Delaunay triangulation* of a given point set.

1.1 Motivation

The concept of property testing can be viewed as a relaxation of a standard decision procedure: instead of deciding whether an object has a given property or not, we only have to distinguish between the case when the object has the property and the case when the object is far from it. Property testing algorithms have been extensively studied in the literature in the recent years (cf. [12, 14, 22] and the references therein).

We believe that property testing is a natural form of approximation for many problems in Computational Geometry. Whether or not the approximation provided by a standard approximation algorithm is more useful than that of property testing depends on the problem at hand, but in some natural situations being close to a property is almost as good as having the property. As a possible motivating example for our study, let us consider the following problem from Computer Graphics of designing efficient *rendering algorithms*. Rendering is the computational task to display a virtual scene on the screen. Such a virtual scene is typically composed of several millions of polygons. Using nowadays graphics hardware and a standard z-buffer algorithm it is not possible to render all polygons in a reasonable time. Therefore, it is necessary to determine in advance if certain parts of the scene cannot be seen from the current point of view. This process is called *occlusion culling*. Unfortunately, it is very difficult to compute exactly those polygons that cannot be seen from a certain point of view. Many approaches have been developed to deal with this problem and some of them work only for objects that have (roughly) a convex shape (see, e.g., the survey [6]). If we could determine quickly in advance whether an object is convex or far from convex then we could quickly decide if it is useful for the purpose of (online) occlusion culling. Such a service can be provided by a property testing algorithm. Therefore we consider the closely related problem, if a point set (possibly the set of vertices of an object used for culling purposes) is in convex position.

1.2 Related works

Despite a large body of research in testing combinatorial properties, there has not been much work done in testing geometric properties. Alon et al. [1] and Czumaj and Sohler [8] investigated property testing of clustering points in $\mathbb{R}^d$. Some basic algorithms testing geometric properties are described in [7] and [9].

In the context of computational geometry a related notion of designing *program checkers* for some basic geometric tasks has been investigated by Mehlhorn et al. [20]. Their approach is to design simple and efficient algorithms which check whether an output of a geometric task is correct. The main difference between

Problem	Query complexity	Lower bound	Running time
disjointness of geometric objects	$\mathcal{O}(\sqrt{n/\epsilon})$	$\Omega(\sqrt{n/\epsilon})$	$T(\mathcal{O}(\sqrt{n/\epsilon}))$
disjointness of polytopes	$\mathcal{O}\left(\frac{d}{\epsilon}\cdot \ln(d/\epsilon)\right)$	$\Omega(1/\epsilon)$	$T(\mathcal{O}\left(\frac{d}{\epsilon}\cdot \ln(d/\epsilon)\right))$
sorted convex polygons	$\mathcal{O}(1/\epsilon)$ deterministic(!)	$\Omega(1/\epsilon)$	$\mathcal{O}(1/\epsilon)$ deterministic(!)
convex position	$\mathcal{O}(\sqrt[d+1]{n^d/\epsilon})$	$\Omega(\sqrt[d+1]{n^d/\epsilon})$	$\min\{T(\mathcal{O}(\sqrt[d+1]{n^d/\epsilon})), \mathcal{O}(n/\epsilon)\}$
Delaunay triangulation (edit distance)	$\mathcal{O}(n)$	$\Omega(n)$	$\mathcal{O}(n)$
Delaunay triangulation (Def. 26)	$\mathcal{O}(1/\epsilon)$	$\Omega(1/\epsilon)$	$\mathcal{O}(1/\epsilon)$

Table 1: ϵ-testers obtained in the paper. $T(m)$ denotes the best known running time of the algorithm deciding given property and we consider lower bounds with respect to the query complexity.

our approach and that in [20] is that property testing algorithms do not aim at deciding the property at hand (in this case, whether the task output is correct), but instead we provide a certain notion of *approximation* — we can argue that we do not accept the property (or the algorithm's output) only if it is far from being satisfied. This relaxation enables us to obtain algorithms with running times *significantly* better than those in [20]. We sacrifice the 100% guarantee for the correctness of the output in the approach in [20] and achieve a *99.9%* guarantee with a much lower complexity.

We notice also that a conceptually similar (but not the same) type of approximation (closeness) as that used in property testing have been also used before in computational geometry community, for example, for geometric pattern matching problems and in [2] in a relaxation of the standard motion planning problem. Also, a similar notion has been studied in the context of *metrology of geometric tolerancing*, see e.g., [21].

Ergün et al. [11] considered a related property that a sequence of points defines a convex hull in $\mathbb{R}^2$. This problem is very different from the problem of testing convex position considered in the current paper. The problem studied in [11] is to test if a *sequence* of points represents a polygon in $\mathbb{R}^2$ defines the convex hull for the points in the sequence and it is possible to design a tester with (asymptotically) the same complexity as that of testing sorting sequences, that is, of $\mathcal{O}(\log n)$. The problem considered in the current paper requires to test if a *set* of points is in convex position and its complexity is provably significantly higher (as we prove in Lemma 20, it is $\Theta(n^{2/3})$ in 2-dimensional case). Furthermore, it is unclear how the techniques from [11] can be applied to d-dimensional problem for $d \geq 3$, while our techniques are extendable also to the high-dimensional variant of the problem. The main reason behind this discrepancy is that in [11], the polygon representation of the input is used, while we are using the geometric representation. Thus the problems considered in [11] contains more structure (and therefore a faster tester exists) but the techniques do not seem to generalize to higher dimension.

1.3 Description of new results

We provide efficient ϵ-testers for some basic problems in computations geometry. We investigate problems that we found most representative for our study and that play an important role in the field. Table 1 summarizes the bounds of our ϵ-tester developed in the paper in details. As it can be seen from Table 1, all our algorithms are asymptotically tight or almost tight regarding their query complexity. We also prove a general lower bound which shows that for many problems no deterministic ϵ-tester exists with the query complexity $o(n)$ already for $\epsilon = \frac{1}{2}$.

All our property testers use the following approach: we first sample at random a sufficiently large subset of input basic objects (e.g., input points) and then analyze the sample to obtain information about the whole input. The key issue in this method is to provide a good description of the input's property using a small input's sample. This approach is used in a simple way in our ϵ-testers for disjointness of geometric objects and Delaunay triangulation, where such a description is easy to show. For disjointness of polytopes and for testing convex position our arguments are more elaborate, because for these two problems it is more complicated to prove the desired property of the whole input out of the analysis of small sample.

Technically, the analysis of testing convex position is most complicated. We develop a property testing algorithm for the convex position property of point sets in $\mathbb{R}^d$ with query complexity of $\mathcal{O}\left(\sqrt[d+1]{n^d/\epsilon}\right)$. The algorithm for an input set P of n points in $\mathbb{R}^d$ tests $\mathcal{O}(\sqrt[d+1]{n^d/\epsilon})$ points and is able to distinguish between a set P in convex position and a set P that is ϵ-*far* from being in convex position. (Here, we say a point set is ϵ-far from being in convex position if one has to modify at least an ϵ-fraction of the points to obtain a set of points in convex position.) The algorithm is randomized and it will always accept a set in convex position but it may fail with probability at most $1/3$ for a set not in convex position. We also show that our property testing algorithm is asymptotically optimal with respect to its query complexity, that is, that every property tester for convex position must query $\Omega(\sqrt[d+1]{n^d/\epsilon})$ points.

We mention also an important feature of all our algorithms, which is that all our testers supply proofs of violation of the property for rejected objects.

Throughout the paper we always assume that d, the dimension of the Euclidean space under consideration, is a constant. Moreover, we assume the size of the input object to be much bigger than d. Throughout the paper we suppose also that all points/arrangements are in general position; this assumption is critical to our analysis and our results hinge on it.

Organization. We begin in Section 2 with preliminaries. Next, in Section 3, we present some basic probabilistic results about random sampling. In Section 4 we develop a tester for disjointness of a pair of V-polytopes. In the next Section 5, we discuss the problem of testing if a point set is in convex position. First, in Section 5.1, we develop a property tester for this problem, and then, in Section 5.2, we prove that our tester has optimal complexity. Section 6 discusses testers for Delaunay triangulations. In Section 7, we provide simple deterministic lower bounds for ϵ-testers. Section 8 contains final remarks.

2 Preliminaries

We begin with some basic notation and definitions. We use the $\widetilde{\mathcal{O}}$-notation to hide polylogarithmic factors, i.e., we have $\widetilde{\mathcal{O}}(n) = \mathcal{O}(n \log^{\mathcal{O}(1)} n)$. We write $[n] = \{1, \dots, n\}$ for the set of integers between 1 and n.

Throughout the paper we assume that our input point set P is in *general position*, i.e., there are no $k+1$ points in P that lie in a k-dimensional affine subspace for $k < d$. The assumption about general position is critical to our analysis: the testers will fail if this assumption is not satisfied.

The goal of property testing is to develop efficient *property testers*. A property tester for Π is an algorithm that gets a distance parameter ϵ and the size n of the input point set P. The property tester has *oracle access* to the input function f (for each $x \in [n]$ it may query for the value of $f(x)$). A property tester must[3]

- accept every function $f \in \Pi$, and
- reject every function f that is ϵ-far from Π with probability at least $\frac{2}{3}$.

[3]We consider a *one-sided error* model, though in the literature also a *two-sided error* model has been considered. In the two-sided error model the goal is to distinguish with probability at least $\frac{2}{3}$ between the case $f \in \Pi$ and of f being ϵ-far from Π.

Notice that *if* $f \notin \Pi$ *and* f *is not* ϵ*-far from* Π*, then the outcome of the algorithm can go either way.*

3 Auxiliary lemmas about random sampling

All our testing algorithms use the following *random sampling* approach: pick a sample set of certain pre-specified size independently and uniformly at random, and then analyze properties of the sample. In this section we present auxiliary probabilistic lemmas that will be used in these analyzes. Although all our results are tuned for the applications presented in this paper, they can be also applied in more general context.

Lemma 1 *Let* Ω *be an arbitrary set of* n *elements. Let* k *and* ℓ *be arbitrary integers (possibly dependent on* n*) and let* s *be an arbitrary integer such that* $s \geq \frac{2n}{(2k)^{1/\ell}}$*. Let* $W_1, \dots, W_k$ *be arbitrary disjoint subsets of* Ω*, each of size* ℓ*. Let* W *be a subset of* Ω *of size* s *which is chosen independently and uniformly at random. Then*

$$\mathbf{Pr}\left[\exists j \in [k] \; : \; (W_j \subseteq W)\right] \;\; \geq \;\; \frac{1}{4} \; .$$

Proof : We first observe that we can focus on the case $k \leq \frac{n}{2}$, because if $k > \frac{n}{2}$, then every set W_i contains exactly one element and then we immediately get $\mathbf{Pr}[\exists j \in [k] \; : \; (W_j \subseteq W)] \geq \frac{1}{2}$. Furthermore, since $k \leq \frac{n}{2}$ yields $\ell + \frac{n-\ell}{(2k)^{1/\ell}} \leq \frac{2n}{(2k)^{1/\ell}}$, it is sufficient to consider only the case $s = \ell + \frac{n-\ell}{(2k)^{1/\ell}}$. Next, by Boole-Benferroni inequality, we obtain

$$\mathbf{Pr}\left[\exists j \in [k] \; : \; W_j \subseteq W\right] \;\geq\; \sum_{j=1}^{k} \mathbf{Pr}[W_j \subseteq W] \;-\; \sum_{1 \leq i < j \leq k} \mathbf{Pr}[(W_i \cup W_j) \subseteq W] \; .$$

Furthermore, we observe that for every $j \in [k]$ it holds that

$$\mathbf{Pr}[W_j \subseteq W] \;=\; \frac{\binom{n-\ell}{s-\ell}}{\binom{n}{s}} \;=\; \frac{(n-\ell)!}{(s-\ell)!\,(n-s)!} \cdot \frac{s!\,(n-s)!}{n!} \;=\; \frac{(n-\ell)!\,s!}{n!\,(s-\ell)!} \;=\; \prod_{r=0}^{\ell-1} \frac{s-r}{n-r} \; .$$

Similar arguments can be used to show that for every $i, j \in [k]$, if $i \neq j$, then it holds that

$$\mathbf{Pr}[(W_i \cup W_j) \subseteq W] \;=\; \frac{\binom{n-2\ell}{s-2\ell}}{\binom{n}{s}} \;=\; \prod_{r=0}^{2\ell-1} \frac{s-r}{n-r} \;=\; \prod_{r=0}^{\ell-1} \frac{s-r}{n-r} \cdot \prod_{r=0}^{\ell-1} \frac{(s-\ell)-r}{(n-\ell)-r} \; .$$

Hence, Boole-Benferroni inequality implies that

$$\begin{aligned}
\mathbf{Pr}\left[\exists j \in [k] \; : \; (W_j \subseteq W)\right] \;&\geq\; k \cdot \prod_{r=0}^{\ell-1} \frac{s-r}{n-r} - \binom{k}{2} \cdot \prod_{r=0}^{\ell-1} \frac{s-r}{n-r} \cdot \prod_{r=0}^{\ell-1} \frac{(s-\ell)-r}{(n-\ell)-r} \\
&=\; k \cdot \prod_{r=0}^{\ell-1} \frac{s-r}{n-r} \cdot \left(1 - \frac{k-1}{2} \cdot \prod_{r=0}^{\ell-1} \frac{(s-\ell)-r}{(n-\ell)-r}\right) \\
&\geq\; k \cdot \prod_{r=0}^{\ell-1} \frac{s-\ell}{n-\ell} \cdot \left(1 - k \cdot \prod_{r=0}^{\ell-1} \frac{s-\ell}{n-\ell}\right) \\
&=\; k \cdot \left(\frac{s-\ell}{n-\ell}\right)^{\ell} \cdot \left(1 - k \cdot \left(\frac{s-\ell}{n-\ell}\right)^{\ell}\right) \; .
\end{aligned}$$

Now, since we assumed that $s = \ell + \frac{n-\ell}{(2k)^{1/\ell}}$, our calculations above yield $\mathbf{Pr}[\exists j \in [k] \,:\, (W_j \subseteq W)] \geq \frac{1}{4}$, and thus the lemma follows. □

Corollary 2 *If in the Lemma 1 we have* $s \geq \frac{8n}{(2k)^{1/l}}$*, then*

$$\mathbf{Pr}\,[\exists j \in [k] \,:\, (W_j \subseteq W)] \geq \frac{2}{3} \enspace .$$

Proof : The proof follows from Lemma 1 by standard amplification arguments. Indeed, we consider set W as 4 independently selected sets $W_1^*, W_2^*, W_3^*, W_4^*$, each of size $\frac{s}{4} \geq \frac{2n}{(2k)^{1/l}}$, and apply Lemma 1 to obtain

$$\mathbf{Pr}\,[\exists j \in [k] \,:\, (X_j \subseteq W)] \;\geq\; 1 - \prod_{i=1}^{4} (1 - \mathbf{Pr}[\exists j \in [k] \,:\, (W_j \subseteq W_i^*)]) \;\geq\; 1 - (3/4)^4 \;\geq\; \frac{2}{3} \enspace .$$

□

We can use similar arguments as that in the proof of Lemma 1 to prove the following result.

Lemma 3 *Let* Ω *be an arbitrary set of* n *elements. Let* k, ℓ, *and* r *be arbitrary integers (possibly dependent on* n*) and let* $s \geq \frac{2n}{(2k)^{1/\ell}}$ *be an integer. Let* $X_1, \dots, X_k$ *be any disjoint subsets of* Ω *of size* l *each. Let* $Y_1, \dots, Y_k$ *be any (not necessarily disjoint) subsets of* $\Omega - \bigcup_{j=1}^{k} X_j$*, each of size at least* r*. Let* S *be a subset of* Ω *of size* s *which is chosen independently and uniformly at random. If* $r \geq \frac{n \ln(8k)}{s}$*, then*

$$\mathbf{Pr}\,[\exists j \in [k] \,:\, ((X_j \subseteq S) \wedge (Y_j \cap S \neq \emptyset))] \geq \frac{1}{8} \enspace .$$

Proof : The proof is similar to that of Lemma 1 and uses Boole-Benferroni inequality. As in the proof of Lemma 1, we can set $s = \ell + \frac{n-\ell}{(2k)^{1/\ell}}$. We first give bounds for $\mathbf{Pr}[(X_j \subseteq S) \wedge (Y_j \cap S \neq \emptyset))]$ and $\mathbf{Pr}[(X_i \cup X_j \subseteq S) \wedge (Y_i \cap S \neq \emptyset) \wedge (Y_j \cap S \neq \emptyset)]$.

Notice that if $i \neq j$, then

$$\mathbf{Pr}[(X_i \cup X_j \subseteq S) \wedge (Y_i \cap S \neq \emptyset) \wedge (Y_j \cap S \neq \emptyset)] \leq \mathbf{Pr}[(X_i \cup X_j \subseteq S)] \enspace .$$

Next, we observe that by the union bound, we obtain that

$$\begin{aligned}
\mathbf{Pr}[(X_j \subseteq S) \wedge (Y_j \cap S \neq \emptyset))] &= 1 - \mathbf{Pr}[(X_j \not\subseteq S) \vee (Y_j \cap S = \emptyset))] \\
&\geq 1 - (\mathbf{Pr}[X_j \not\subseteq S] + \mathbf{Pr}[Y_j \cap S = \emptyset]) \\
&= \mathbf{Pr}[X_j \subseteq S] - \mathbf{Pr}[Y_j \cap S = \emptyset] \enspace .
\end{aligned}$$

Therefore, by Boole-Benferroni inequality we obtain that

$$\begin{aligned}
&\mathbf{Pr}\,[\exists j \in [k] \,:\, ((X_j \subseteq S) \wedge (Y_j \cap S \neq \emptyset))] \;\geq \\
&\quad\geq \sum_{j \in [k]} \mathbf{Pr}[(X_j \subseteq S) \wedge (Y_j \cap S \neq \emptyset))] - \sum_{i,j \in [k], i \neq j} \mathbf{Pr}[(X_i \cup X_j \subseteq S) \wedge (Y_i \cap S \neq \emptyset) \wedge (Y_j \cap S \neq \emptyset)] \\
&\quad\geq \left(\sum_{j \in [k]} \mathbf{Pr}[X_j \subseteq S] - \sum_{i,j \in [k], i \neq j} \mathbf{Pr}[(X_i \cup X_j \subseteq S)] \right) - \sum_{j \in [k]} \mathbf{Pr}[Y_j \cap S = \emptyset] \enspace .
\end{aligned}$$

Furthermore, by our arguments from the proof of Lemma 1, we know that for our choice of s we have,

$$\left(\sum_{j \in [k]} \mathbf{Pr}[X_j \subseteq S] - \sum_{i,j \in [k], i \neq j} \mathbf{Pr}[(X_i \cup X_j \subseteq S)] \right) \geq \frac{1}{4} .$$

Therefore, we only have to focus on giving an upper bound for $\mathbf{Pr}[Y_j \cap S = \emptyset]$. Then we have

$$\begin{aligned} \mathbf{Pr}[Y_j \cap S = \emptyset] &\leq \frac{\binom{n-r}{s}}{\binom{n}{s}} = \prod_{t=0}^{r-1} \frac{n-s-t}{n-t} = \prod_{t=n-r+1}^{n} \left(1 - \frac{s}{t}\right) \leq \exp\left(\sum_{t=n-r+1}^{n} \frac{-s}{t} \right) \\ &= \exp\left(-s \cdot \sum_{t=n-r+1}^{n} \frac{1}{t} \right) \leq \exp\left(-s \cdot r \cdot \frac{1}{n} \right) . \end{aligned}$$

Finally, we observe that since we assumed that $r \geq \frac{n \ln(8k)}{s}$, we obtain $\mathbf{Pr}[Y_j \cap S = \emptyset] \leq \frac{1}{8k}$. Hence, we obtain the claimed bound:

$$\begin{aligned} \mathbf{Pr}\left[\exists j \in [k] \,:\, ((X_j \subseteq S) \wedge (Y_j \cap S \neq \emptyset))\right] &\geq \left(\sum_{j \in [k]} \mathbf{Pr}[X_j \subseteq S] - \sum_{i,j \in [k], i \neq j} \mathbf{Pr}[(X_i \cup X_j \subseteq S)] \right) \\ &\quad - \sum_{j \in [k]} \mathbf{Pr}[Y_j \cap S = \emptyset] \geq \frac{1}{4} - k \cdot \frac{1}{8k} = \frac{1}{8} . \end{aligned}$$

□

The following corollary follows immediately from Lemma 3 by setting the parameters appropriately and by amplification.

Corollary 4 *Let Ω be an arbitrary set of n elements. Let d be an arbitrary integer and let $J = \frac{\epsilon n}{d+1}$ be an integer for certain real ϵ, $0 < \epsilon < 1$. Let $W_1, \dots, W_J$ be disjoint subsets of Ω of size $d+1$ each. Further, let $U_1, \dots, U_J$ be arbitrary (not necessarily disjoint) subsets of $\Omega - \bigcup_{j=1}^{J} W_j$, each of size at least $8 \cdot (\epsilon n)^{1/(d+1)} \cdot \ln(\frac{\epsilon n}{d+1})$. Let S be a subset of Ω of size at least $36 \cdot (n^d/\epsilon)^{1/(d+1)}$ which is chosen independently and uniformly at random. Then*

$$\mathbf{Pr}\left[\exists j \in [J] \; \exists u \in U_j \,:\, (W_j \cup \{u\} \subseteq S)\right] \geq \frac{2}{3} .$$

Next, we prove that the bounds for s in Lemma 1 and in Lemma 3 are tight up to a constant factor.

Lemma 5 *Let Ω be an arbitrary set of n elements. Let k, ℓ, and s be arbitrary integers (possibly dependent on n). Let $W_1, \dots, W_k$ be any disjoint subsets of Ω of size ℓ each. Let S be a subset of Ω of size s which is chosen independently and uniformly at random. For every real p, $0 < p < 1$, (possibly depending on other parameters), if $s \leq (n - (\ell - 1)) \cdot (p/k)^{1/\ell}$, then*

$$\mathbf{Pr}\left[\exists j \in [k] \,:\, (W_j \subseteq S)\right] \leq p .$$

Proof : We use the union bound to obtain that

$$\mathbf{Pr}\left[\exists j \in [k] \,:\, (W_j \subseteq S)\right] \leq \sum_{j=1}^{k} \mathbf{Pr}[W_j \subseteq S] = \sum_{j=1}^{k} \prod_{r=0}^{\ell-1} \frac{s-r}{n-r} \leq k \cdot \left(\frac{s}{n - (\ell - 1)} \right)^{\ell} .$$

Therefore, if $s \leq (n - (\ell - 1)) \cdot (p/k)^{1/\ell}$, then the above inequality is upper bounded by p. □

3.1 Sample application: Disjointness of generic geometric objects

To present the flavor of our property testing algorithms and their use of the lemmas above, we show now a simple tester that verifies whether a collection of objects is disjoint (this result appears also in [10]). The problem of deciding whether an arrangement of objects is intersection-free belongs to the most fundamental problems in computational geometry. A typical example is to verify whether a set of line segments in the plane or a set of hyper-rectangles or polytopes in $\mathbb{R}^d$ is intersection free. We assume the oracle allows to query for the input objects and we suppose that there is an algorithm A that solves the exact decision problem of testing whether a set of objects is disjoint. Furthermore, we consider the problem only for *generic objects* (i.e., we use no information about the structure of the objects used, and so, the object analyzed may be of arbitrary complexity; observe, however, that the running time complexity of this problem depends heavily on the object properties, because so does the (implicitly used) algorithm A).

We say a collection of objects $\mathcal{X}$ is *ϵ-far from being disjoint* if there is no set $T \subseteq \mathcal{X}$ with $|T| \leq \epsilon \cdot |\mathcal{X}|$ such that $\mathcal{X} - T$ is disjoint. Then the following algorithm is a simple ϵ-tester:

```
DISJOINTNESS (set $\mathcal{X}$ consisting of $n$ objects)
    Choose a set $S \subseteq \mathcal{X}$ of size $8\sqrt{n/\epsilon}$ uniformly at random
    Check whether $S$ is disjoint using algorithm $A$
    **if** $S$ is disjoint **then** *accept*
    **else** *reject*
```

Theorem 6 *Algorithm* DISJOINTNESS($\mathcal{X}$) *is an ϵ-tester with the query complexity $\Theta(\sqrt{n/\epsilon})$. Furthermore, its query complexity is asymptotically* optimal. *If the running time of the algorithm A for k objects is $T(k)$, then the running time of* DISJOINTNESS($\mathcal{X}$) *is $\mathcal{O}(T(8\sqrt{n/\epsilon}))$.*

Proof : We first prove that DISJOINTNESS($\mathcal{X}$) is an ϵ-tester. Clearly, if $\mathcal{X}$ is intersection-free, then the algorithm accepts $\mathcal{X}$. So let us suppose that $\mathcal{X}$ is ϵ-far from being intersection-free. It easy to see that case we can apply $\frac{1}{2} n \epsilon$ times the following procedure to $\mathcal{X}$: pick a *pair of intersecting objects* and remove it from $\mathcal{X}$. Thus in order to prove that DISJOINTNESS($\mathcal{X}$) is an ϵ-tester it is sufficient to show that with probability at least $\frac{2}{3}$, at least one of the pairs is chosen to S (because the objects from each pair intersect each other). To conclude the bound we apply Corollary 2. Sets $W_1, \dots, W_k$ (with $k = \frac{1}{2} \epsilon n$) in Corollary 2 are the aforementioned pairs of intersecting objects and $W = S$. Since the size of S is $8\sqrt{n/\epsilon}$, the upper bound for the query complexity follows.

To prove that every ϵ-tester has query complexity of $\Omega(\sqrt{n/\epsilon})$, let us consider two possible sets $\mathcal{X}$: the one in which $\mathcal{X}$ is pairwise disjoint and another one in which $\mathcal{X}$ consists of ϵn pairs of objects intersecting each other within the pair and $n - 2\epsilon n$ objects, each having no intersection with any other object. Then, any ϵ-tester which distinguishes between the first and the second input and rejects the second set must query the oracle to obtain at least once two objects from the same pair. Hence, we can apply Lemma 5 to show that in order the tester to reject $\mathcal{X}$ the oracle must be queried at least $\Omega(\sqrt{n/\epsilon})$ times.

The last claim regarding the running time of the algorithm follows from our discussion above. □

4 Disjointness of V-Polytopes

For any point set P in $\mathbb{R}^d$ let $\mathcal{CH}(P)$ denote the convex hull of P. In this section we consider a special case of testing disjointness of geometric objects and analyze the property if for a given point sets (in convex

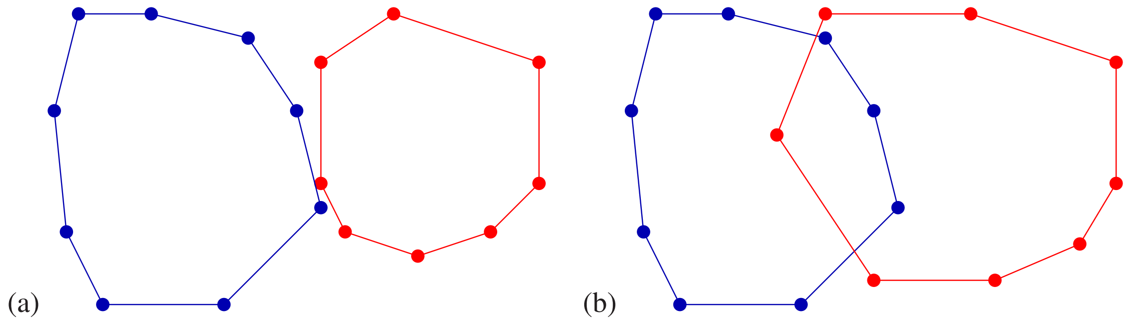

Figure 1: (a) Two V-polygons defined by the two sets of points that are disjoint and (b) two V-polygons that are not disjoint.

positions) R and B, the two polytopes $\mathcal{CH}(R)$ and $\mathcal{CH}(B)$ (so called *V-polytopes*[4]) are disjoint. We refer to the point sets R and B as red and blue, respectively. Let $P = R \cup B$ and $n = |P|$.

Definition 7 *Two polytopes $\mathcal{CH}(R)$ and $\mathcal{CH}(B)$ with finite points sets $R, B \subseteq \mathbb{R}^d$ in convex position are* ϵ-far from being disjoint, *if there is no set $V \subseteq R \cup B$, $|V| \le \epsilon \cdot |R \cup B|$, such that $\mathcal{CH}(R - V)$ and $\mathcal{CH}(B - V)$ are disjoint.*

In this section we show that the following algorithm is an ϵ-tester for disjointness of V-polytopes.

```
DISJOINTNESS (R, B):
  Choose a set $S \subseteq P$ of size $\Theta\left(\frac{d}{\epsilon}\ \ln(d/\epsilon)\right)$ uniformly at random
  Test whether $R \cap S$ and $B \cap S$ are disjoint
  **if** $R \cap S$ and $B \cap S$ are disjoint **then** *accept*
  **else** *reject*
```

Before we proceed with the analysis of Algorithm DISJOINTNESS, we first recall a result on ε-nets due to Haussler and Welzl [18]. Consider the range space $(\mathbb{R}^d, \mathcal{H})$ with $\mathcal{H}$ being the set of all halfspaces in $\mathbb{R}^d$. It is known that $\mathcal{H}$ has Vapnik-Chervonenkis-dimension $d + 1$. For any real $0 < \varepsilon < 1$ and any finite set $\mathcal{X} \subseteq \mathbb{R}^d$, let us consider the set $\mathcal{H}_{\mathcal{X},\varepsilon}$ of all $h \in \mathcal{H}$ which contain a fraction of the points in $\mathcal{X}$ of size greater than or equal to ε, i.e., $\mathcal{H}_{\mathcal{X},\varepsilon} = \{h \in \mathcal{H} \ : \ |h \cap \mathcal{X}| \ge \varepsilon \cdot |\mathcal{X}|\}$. An ε-*net* of $\mathcal{X}$ for the range space $(\mathbb{R}^d, \mathcal{H})$ is a subset $\mathcal{Y} \subseteq \mathcal{X}$ containing a point in each $h \in \mathcal{H}_{\mathcal{X},\varepsilon}$. The celebrated result of Haussler and Welzl [18] on the existence of ε-nets states that a random subset $\mathcal{Z}$ of $\mathcal{X}$ of size at least $c\ \frac{d}{\varepsilon}\ \ln(d/\varepsilon)$ with probability at least 0.9, where c is a suitable chosen absolute constant, is an ε-net of $\mathcal{X}$. Now, we show how this result can be applied to design an ϵ-tester for disjointness of V-polytopes with $\epsilon = 2\,\varepsilon$.

We begin with a simple observation (which follows easily from Definition 7) on the structure of sets ϵ-far from being disjoint. For any hyperplane h we denote by h^- and h^+ two halfspaces induced by h. It is easy to see that two convex polytopes are disjoint if and only if the can be separated by some hyperplane. The following result extends this observation to pairs of sets that are ϵ-far from being disjoint.

Observation 8 *Let R and B be two point sets whose V-polytopes are ϵ-far from being disjoint, and let h be any hyperplane. It holds that $|R \cap h^+| + |B \cap h^-| \ge \epsilon\, n$ and $|R \cap h^-| + |B \cap h^+| \ge \epsilon\, n$. Therefore, in particular, at least one of the following four cases must hold:*

[4]For any point set X of points in $\mathbb{R}^d$ that are in *convex position* (cf. Definition 11), a *V-polytope* is its convex hull $\mathcal{CH}(X)$.

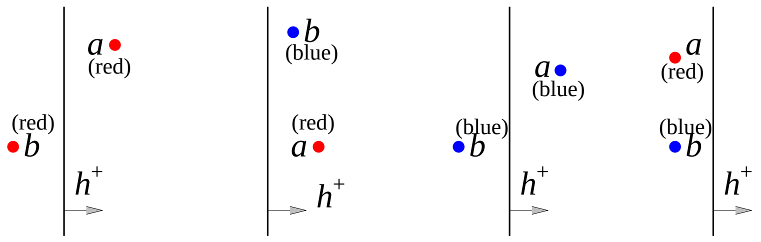

Figure 2: Different types of witnesses for which hyperplane h does not separate $\mathcal{CH}(R)$ from $\mathcal{CH}(B)$.

(1) $|R \cap h^-|, |R \cap h^+| \geq \epsilon n/2$,

(2) $|R \cap h^-|, |B \cap h^-| \geq \epsilon n/2$,

(3) $|R \cap h^+|, |B \cap h^+| \geq \epsilon n/2$,

(4) $|B \cap h^-|, |B \cap h^+| \geq \epsilon n/2$. □

With this simple observation and the result above on ε-nets, we can prove the following theorem.

Theorem 9 *Algorithm* DISJOINTNESS *is an ϵ-tester for disjointness of V-polytopes with the query complexity $\mathcal{O}\left(\frac{d}{\epsilon}\ln(d/\epsilon)\right)$.*

Proof : Since the query complexity follows immediately from the bound on the size of S, we focus on showing that DISJOINTNESS is a proper ϵ-tester. It is easy to see that if R and B are disjoint then DISJOINTNESS always accepts the input. So let us suppose that the input is ϵ-far from being disjoint. Since $\mathcal{CH}(R)$ and $\mathcal{CH}(B)$ are disjoint if and only if they can be separated by some hyperplane, our goal is to ensure that with probability at least $\frac{2}{3}$, for every hyperplane h, the sample set S contains a witness that h does not separate $\mathcal{CH}(R)$ from $\mathcal{CH}(B)$. It is easy to see that such a witness exists if we could ensure that for every hyperplane h set S contains two points a, b such that either (a, b) or (b, a) belongs to one of the following four sets (see also Figure 2):

$$(R \cap h^-) \times (R \cap h^+) \ , \quad (R \cap h^-) \times (B \cap h^-) \ , \quad (R \cap h^+) \times (B \cap h^+) \ , \quad (B \cap h^-) \times (B \cap h^+) \ .$$

Moreover, by Observation 8 we know that for at least one of these four sets, the sets in the Cartesian product are of cardinality at least $\epsilon n/2$ each. Let such two sets be called a *representative* for h. Therefore, by our discussion above, in order to complete the proof of the theorem we only must show that S intersects each representative set for *every* hyperplane h in $\mathbb{R}^d$. And this follows from the result on the randomized construction of ε-nets mentioned above (with $\epsilon = 2\,\varepsilon$). Indeed, by the aforementioned result due to Haussler and Welzl [18], the set S is with probability at least $\frac{2}{3}$ an $(\epsilon/2)$-net of each of R and B for the range space $\left(\mathbb{R}^d, \mathcal{H}\right)$. Therefore, by the definition of ε-nets, S intersects every representative of each hyperplane h in $\mathbb{R}^d$. □

One can also easily improve the query complexity of algorithm DISJOINTNESS in the special case when $d = 2$ *and* the input polygons are stored in a *sorted array* (i.e., the input points of each of R and B are stored in an array such that the consecutive vertices of the polygon $\mathcal{CH}(R)$ ($\mathcal{CH}(B)$, respectively) are on the consecutive positions in the array).

Theorem 10 *For all pairs of polygons in $\mathbb{R}^2$ represented by a sorted array there exists a deterministic ϵ-tester for disjointness of V-polytopes with the query complexity and the running time of $\mathcal{O}(1/\epsilon)$.*

Proof : Using similar arguments as those in Observation 8, one can show that if we take as S every $\frac{\epsilon n}{2}$th element from each polygon R and B, then if R and B are ϵ-far from being disjoint, then there will be a certificate for that in S, i.e., $S \cap \mathcal{CH}(R)$ and $S \cap \mathcal{CH}(R)$ will intersect. □

5 Testing convex position

In this section, we present a detailed analysis of the problem of testing if a given set of points in $\mathbb{R}^d$ is in convex position. First, in Section 5.1, we develop a simple property tester and then prove its properties. Next, in Section 5.2, we give a matching lower bound for the complexity of any property tester for convex position. We will summarize our discussion in Section 5.3.

We begin with some basic definitions used in our analysis.

Definition 11 *A point $p \in P$ is called an* extreme point *of P if p is a vertex of $\mathcal{CH}(P)$, the convex hull of P. The interior of $\mathcal{CH}(P)$ will be denoted by $\mathcal{CH}_{\mathrm{int}}(P)$. A point set P is in* convex position *if every point in P is an extreme point, that is, $\mathcal{CH}_{\mathrm{int}}(P) \cap P = \emptyset$.*

We assume that P is represented by a function $f : [n] \to \mathbb{R}^d$. We also need a distance measure that determines when a point set is far from having the convex position property. We use the most natural definition (most typical for property testing applications) which measures the fraction of the input that has to be changed to obtain an object having the property.

Definition 12 *A set P of n points is ϵ-far from convex position if no set Q of size (at most) ϵn exists such that $P \setminus Q$ is in convex position.*

5.1 Testing convex position — upper bound

In this section we prove that the following algorithm is a property tester for convex position:

```
CONVEXTESTER $(P, \epsilon)$
    let $s = 16\left(4\sqrt[d+1]{n^d/\epsilon} + 2d + 2\right)$
    Choose a set $S \subseteq P$ of size $s$ uniformly at random
    if $S$ is in convex position then accept
    else reject
```

In order to show that CONVEXTESTER is a property tester we have to prove that (1) it accepts every point set in convex position and (2) it rejects every point set that is ϵ-far from convex position with probability at least $2/3$.

We observe that CONVEXTESTER accepts every point set in convex position because every subset of a set in convex position is in convex position, as well. Thus we only have to prove that a point set that is ϵ-far from convex position is rejected with high probability.

Now let us assume that P is ϵ-far from convex position. We want to prove that algorithm CONVEXTESTER rejects P with probability at least $2/3$. We will use the Carathéodory's Theorem [4] that implies that every point set that is not in convex position can be rejected by finding a subset of $d+2$ points that is not in convex position. We refer to such a subset as a *counter example*.

The idea of the analysis is now to consider sets W_i, $1 \le i \le \frac{\epsilon n}{2(d+1)}$, of $d+1$ points that can be easily extended to a set of $d+2$ points that is not in convex position. Easily extendible means that there is a large set U_i of points such that for each $p \in U_i$ the set $W_i \cup \{p\}$ is a counter example.

Finding counter examples. We now formalize this idea and prove two technical lemmas.

Lemma 13 *Let P be a set of n points in $\mathbb{R}^d$. Let $p \in \mathbb{R}^d$ be a point with $p \in \mathcal{CH}_{int}(P)$, where $P \cup \{p\}$ is in general position. Then there exist sets $W \subseteq P$ and $U \subseteq P \setminus W$ with $|W| = d$ and $|U| \ge \frac{n}{d+1}$ such that $p \in \mathcal{CH}_{int}(W \cup \{q\})$ for each $q \in U$.*

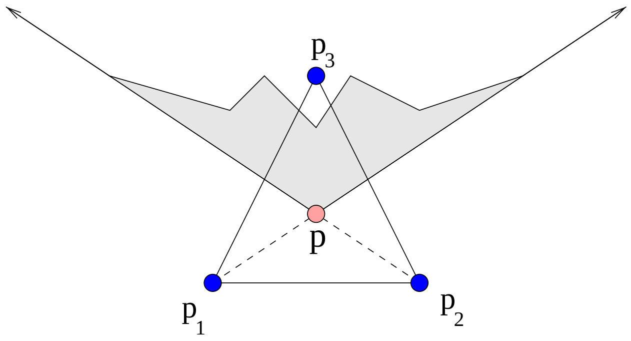

Figure 3: Illustration for the proof of Lemma 13. One of the cones C_i (this one is defined by p_1 and p_2).

Proof : Since $p \in \mathcal{CH}_{int}(P)$, by Carathéodory's Theorem [4] and by the general position assumption, it follows that there is a set $W^* \subseteq P$ of size $d+1$ such that $p \in \mathcal{CH}_{int}(W^*)$. Let us denote by W_i^*, $1 \le i \le d+1$, the subsets of W^* of size d. We show that one of the subsets W_i^* of W^* satisfies Lemma 13. Without loss of generality, let us assume that $p = (0, \dots, 0)$. We partition $\mathbb{R}^d$ into $d+1$ cones. Let W_i^-, $1 \le i \le d+1$, denote the set of points $\{(-x_1, \dots, -x_d) : (x_1, \dots, x_d) \in W_i^*\}$. The conic combination of the point vectors in the set W_i^- defines a cone C_i ,$1 \le i \le d+1$. The union of these cones C_i covers $\mathbb{R}^d$. Thus there is a cone C_j, $1 \le j \le d+1$, that contains at least $\frac{n}{d+1}$ points of P (and also of $P \setminus W_j^*$ since $W_j^* \cap C_j = \emptyset$). We also observe that for each $q \in P \cap C_j$ it holds that $p \in \mathcal{CH}_{int}(W_j^* \cup \{q\})$. Therefore, we can conclude that the sets $W = W_j^*$ and $U = P \cap C_j$ satisfy the lemma. □

The second technical lemma shows that if a point set is ϵ-far from convex position then there are many sets W_i and U_i such that $W_i \cup \{q\}$ is not in convex position for every $q \in U_i$.

Lemma 14 *Let P be a set of n points in $\mathbb{R}^d$ that is ϵ-far from convex position and let $k = \frac{\epsilon n}{d+1}$. Then there exist sets $W_i \subseteq P$, $i \in [k]$, and sets $U_i \subseteq P$, $i \in [k]$, such that the following properties are fulfilled:*

(1) $|W_i| = d+1$, $i \in [k]$,

(2) $W_i \cap W_j = \emptyset$ *for all* $i, j \in [k]$ *with* $i \ne j$,

(3) $W_i \cup \{q\}$ *is not in convex position for every* $q \in U_i$, *and*

(4) $|U_i| \ge (1-\epsilon) \cdot \frac{n}{d+1}$.

Proof : We construct point sets $P_1, P_2, \ldots, P_k$ with $P_1 = P$, $W_i \subseteq P_i$ and $P_{i+1} = P_i \setminus W_i$. Clearly, this definition ensures that the sets W_i are disjoint. We show how to construct sets W_i and P_i.

First of all, observe that for every $i \in [k]$ it holds that $|P_i| = n - (d+1)(i-1)$ and thus $|P_i| \geq n - (d+1)(\frac{\epsilon n}{d+1} - 1)$. This implies that $|P \setminus P_i| < \epsilon n$ and so the set P_i cannot be in convex position (by the definition of ϵ-far we can delete every set of size at most ϵn and do not obtain a point set in convex position). Since P_i is not in convex position there is a point $p \in \mathcal{CH}_{int}(P_i)$. We apply Lemma 13 with p and $P_i \setminus \{p\}$ and obtain the sets W and U. We choose $W_i = W \cup \{p\}$ and $U_i = U$. By Lemma 13, we get $|W_i| = d+1$ and $W_i \cup \{q\}$ is not in convex position for each $q \in U_i$. We already observed that $|P_i| \geq n - (d+1)(\frac{\epsilon n}{d+1} - 1)$. Thus we know that $|P_i \setminus \{p\}| \geq n - \epsilon n$ which means that $|U_i| \geq \frac{n - \epsilon n}{d+1} = (1-\epsilon) \cdot \frac{n}{d+1}$. All properties in Lemma 14 are satisfied and the lemma is proven. □

Lemma 15 *Algorithm* CONVEXTESTER *is a property tester for the convex position property. Its query complexity is* $\mathcal{O}(\sqrt[d+1]{n^d/\epsilon})$.

Proof : We have to prove that CONVEXTESTER accepts every point set in convex position and rejects every point set that is ϵ-far from convex position with probability $2/3$.

We already observed that algorithm CONVEXTESTER accepts every point set in convex position. Therefore, we have to show that every point set that is ϵ-far from convex position is rejected with probability at least $2/3$. Let P be a point set that is ϵ-far from convex position and let $k = \frac{\epsilon n}{d+1}$. Then there exist sets W_i and U_i, $i \in [k]$, with the properties stated in Lemma 14. Now let $S \subseteq P$ be a point set of size $4\sqrt[d+1]{n^d/\epsilon} + 2(d+1)$ chosen uniformly at random from P. Without loss of generality, let us assume that $\epsilon < 1/2$. We view S as the union of two sets W and U of size $4\sqrt[d+1]{n^d/\epsilon}$ and $2(d+1)$, respectively. Clearly, we can assume that each of them is chosen uniformly (and independently) at random from P. We first observe that for a fixed U_i of size $(1-\epsilon)\frac{n}{d+1}$:

$$\mathbf{Pr}[U_i \cap U \neq \emptyset] \;\geq\; 1 - \left(1 - \frac{|U_i|}{n}\right)^{|U|} \geq 1 - 1/e \geq 1/2 \;.$$

Moreover, we know that:

$$\begin{aligned} \mathbf{Pr}[\text{P is rejected}] &\geq \mathbf{Pr}[\exists i \in [k] : W_i \subseteq S \text{ and } U_i \cap S \neq \emptyset] \\ &\geq \mathbf{Pr}[\exists i \in [k] : W_i \subseteq W \text{ and } U_i \cap U \neq \emptyset] \\ &\geq \mathbf{Pr}[\exists i \in [k] : W_i \subseteq W] \cdot 1/2 \;. \end{aligned}$$

Now we use our auxiliary Lemma 1 showing that if we take a sample of size s from a set Ω of n elements, then with good probability we have at least one of k disjoint sets (each of size ℓ) in our sample. To derive a bound on $\mathbf{Pr}[\exists i \in [k] : W_i \subseteq W]$, we apply Lemma 1 with $k = \frac{\epsilon n}{d+1}$, $\ell = d+1$, and

$$s \;=\; 4\sqrt[d+1]{n^d/\epsilon} \;\geq\; \sqrt[d+1]{(2n)^d \cdot (d+1)/\epsilon} \;\geq\; \frac{2n}{(2k)^{1/\ell}}$$

and obtain:

$$\mathbf{Pr}[\exists i \in [k] : W_i \subseteq W] \;\geq\; 1/4 \;.$$

We conclude that

$$\mathbf{Pr}[\text{P is rejected}] \;\geq\; 1/8 \;.$$

If algorithm CONVEXTESTER chooses a set S of size $16 \cdot \left(4 \sqrt[d+1]{n^d/\epsilon} + 2(d+1)\right)$ then

$$\mathbf{Pr}\,[\text{P is rejected}] \;\geq\; 1-(1-1/8)^{16} \;\geq\; 2/3\;,$$

which proves the query complexity stated in Lemma 15. □

5.2 Testing convex position — lower bound for query complexity

In this section we prove that the property tester described in the previous section is asymptotically optimal with respect to the query complexity. We first show that the existence of a property tester with query complexity $q(\epsilon, n)$ implies that there is a *canonical property tester* with the same query complexity, i.e., a property tester that samples a set S of $q(\epsilon, n)$ points uniformly at random and accepts if and only if the sample set S is in convex position. Then we show that for sufficiently large n there is a point set P that is ϵ-far from convex position such that

$$\mathbf{Pr}[\text{S is in convex position }] > 1/3$$

holds, if $S \subseteq P$ is a set of $q(\epsilon, n) = o(\sqrt[d+1]{n^d/\epsilon})$ points chosen uniformly at random from P. This implies that there is no property tester with query complexity $q(\epsilon, n)$.

In this section we consider only properties of point sets that do not depend on their representation as a function $f : [n] \to \mathbb{R}^d$, i.e., either every functional representation of a point set P has the property or no representation has the property. We call these properties *combinatorial properties*.

5.2.1 Canonical property testers

We begin our analysis with a fundamental lemma that is needed for lower bound constructions. We show that if there is a property tester for a combinatorial property Π of point sets with query complexity $q(\epsilon, n)$ then there is a property tester for Π that picks a sample set $S \subseteq P$ of size $q(\epsilon, n)$ uniformly at random and decides based on S and its internal coin flips. The proof is similar to the proof of Lemma 4.1 in [16] which is based on an analogous statement proven by Bar-Yossef et al. in [3].

The idea of the proof is simple: There are $n!$ different representations of the same point set, one for each permutation of $[n]$. When we start our algorithm on a random representation of the same set then choosing the next index adaptively is essentially equivalent to picking another element uniformly at random. This is formalized in the following lemma.

Lemma 16 (Testing by Canonical Property Testers) *Let Π be an arbitrary combinatorial property of point sets and let $\mathbb{A}$ be an arbitrary property tester for Π with query complexity $q(\epsilon, n)$. Then there exists a property tester for property Π that selects a set S of $q(\epsilon, n)$ points uniformly at random and accepts or rejects the input solely on the base of S and its internal coin flips.*

Proof : Recall that a point set $P = \{p_1, \cdots, p_n\}$ in the $\mathbb{R}^d$ is represented by a function $f : [n] \to \mathbb{R}^d$ with $f(i) = p_i$. Since Π is a combinatorial property we know that we can analyze properties of P rather than properties of the function representing P. That is, if one representation (permutation of points) of P has property Π then every other representation also has property Π. We use this fact in the following to construct a property tester that samples uniformly at random from an arbitrary property tester for Π. Both property testers have the same query complexity.

Let $\mathbb{A}$ be an arbitrary property tester for Π. Without loss of generality, we assume that algorithm $\mathbb{A}$ operates in iterations. In each iteration, depending on its internal coin flips and the answers obtained in the past iterations, the algorithm selects a new index i and makes a query for the position of the point p_i.

Now we obtain a new algorithm $\mathbb{A}'$ from $\mathbb{A}$ in the following way: When $\mathbb{A}'$ is started with input point set P of size n (given to the oracle) it first chooses a permutation π of $[n]$ uniformly at random. Then algorithm $\mathbb{A}$ is invoked with oracle access to the point set $\pi(P)$ that is represented by the function $f_\pi : [n] \to \mathbb{R}^d$ defined by $f_\pi(i) = f(\pi(i))$. We observe that $\pi(P)$ has property Π if and only if P has property Π since $\pi(P) = P$ and since property Π is combinatorial. Furthermore, we know that $\mathbb{A}$ is a property tester for property Π. Thus it must accept the input if $\pi(P)$ has property Π. It follows that if P has property Π then $\mathbb{A}'$ accepts.

If P is ϵ-far from Π then so is $\pi(P)$. By the fact that $\mathbb{A}$ is a property tester it follows that $\pi(P)$ is rejected by $\mathbb{A}$ with probability at least $2/3$. It follows that $\mathbb{A}'$ also rejects P with probability at least $2/3$. Thus we know that $\mathbb{A}'$ is a property tester for property Π.

Our next step is to show that in each iteration of algorithm $\mathbb{A}'$ the next chosen point is uniformly distributed among all possible choices (among all points that have not been chosen in a previous iteration). For every sequence r of coin flips let $\mathbb{A}_r$ denote the deterministic algorithm obtained from $\mathbb{A}$ by fixing the outcome of the coin flips according to r. Furthermore, let $\mathbb{A}'_r$ denote the algorithm similar to $\mathbb{A}'$ with the exception that $\mathbb{A}_r$ is invoked instead of $\mathbb{A}$. We show that for every sequence r of coin flips and for every possible sequence of queries and answers the choice of the point selected next by algorithm $\mathbb{A}'$ is uniformly distributed among the points not selected so far.

Let $i^\pi_{1,r}, \ldots, i^\pi_{j,r}$ denote the indices selected by $\mathbb{A}'_r$ in iterations 1 to j, i.e., the indices selected when $\mathbb{A}_r$ is invoked with oracle access to the point set $\pi(P)$. Furthermore, let $q^\pi_{1,r}, \ldots, q^\pi_{j,r}$ denote the points selected by $\mathbb{A}'_r$ in iterations 1 to j, i.e., $q^\pi_{\ell,r} = f_\pi(i^\pi_{\ell,r})$ for $1 \le \ell \le j$. The choice of the index $i^\pi_{j+1,r}$ selected by $\mathbb{A}'_r$ in iteration $j+1$ depends only on the previously selected points $q^\pi_{1,r}, \ldots, q^\pi_{j,r}$. Thus when we condition π by $f_\pi(i^\pi_{\ell,r}) = q^\pi_{\ell,r}$, $1 \le \ell \le j$, the choice of index $i^\pi_{j+1,r}$ is deterministic. Furthermore, we prove in the following claim that for every fixed index $i^\pi_{j+1,r}$ under the same conditioning the point $f_\pi(i^\pi_{j+1,r})$ is uniformly distributed among the points not selected so far:

Claim 17 *Let $i_1, \ldots, i_j \in [n]$ be distinct indices and let $q_1, \ldots, q_j \in P$ be distinct points. Then for each $i \in [n] \setminus \{i_1, \ldots, i_j\}$ and each $p \in P \setminus \{q_1, \ldots, q_j\}$:*

$$\mathbf{Pr}[f_\pi(i) = p \mid f_\pi(i_\ell) = q_\ell, 1 \le \ell \le j] \quad = \quad \frac{1}{n-j} \ .$$

Proof : The number of permutations of $[n]$ with $f_\pi(i_\ell) = q_\ell$ for $1 \le \ell \le j$ is equal to $(n-j)!$. For each $i \in [n] \setminus \{i_1, \ldots, i_j\}$ and each $p \in P \setminus \{q_1, \ldots, q_j\}$ the number of permutations of $[n]$ with $f_\pi(i) = p$ and $f_\pi(i_\ell) = q_\ell$ for $1 \le \ell \le j$ is equal to $(n-j-1)!$. Therefore,

$$\mathbf{Pr}[f_\pi(i) = p \mid f_\pi(i_\ell) = q_\ell, 1 \le \ell \le j] \quad = \quad \frac{(n-j-1)!}{(n-j)!} \quad = \quad \frac{1}{n-j} \ .$$

□

Thus we know that in each iteration of algorithm $\mathbb{A}'_r$ the next chosen point is uniformly distributed among the points not chosen so far. We use this fact to build an algorithm $\mathbb{A}''$ that has the same output distribution as $\mathbb{A}'$. Then we design an algorithm $\mathbb{A}'''$ that samples a set $S \subseteq P$ of $q(\epsilon, n)$ points uniformly at random and then decides based on internal coin flips and the point positions. We observe that we can execute algorithm $\mathbb{A}'$ by first choosing a sequence r of coin flips uniformly at random and then invoking algorithm $\mathbb{A}'_r$. Algorithm $\mathbb{A}''$ works in a similar manner. It first chooses a sequence r of coin flips uniformly at random. Then it invokes algorithm $\mathbb{A}''_r$. Algorithm $\mathbb{A}''_r$ behaves similarly to algorithm $\mathbb{A}'_r$ with the exception that in each iteration when $\mathbb{A}'_r$ requests the position of the point with index i algorithm $\mathbb{A}''_r$ selects uniformly at random an index not selected so far and returns the corresponding point. Let us formalize this point and look at iteration j of both algorithms. Let i_j denote the index selected by algorithm A'_r in iteration j and let i'_j denote the index selected by algorithm A''_r. When during iteration $j+1$ algorithm $\mathbb{A}'_r$ requests the position of the point stored at index i_{j+1} then $\mathbb{A}''_r$ selects an index i'_{j+1} uniformly at random from the set $[n] \setminus \{i'_1, \ldots, i'_j\}$ and answers the query with the position of point i'_{j+1}.

Claim 18 *Let* P *be any point set in the* $\mathbb{R}^d$. *If algorithms* $\mathbb{A}''$ *and* $\mathbb{A}'$ *are invoked with oracle access to the same function representing* P *then their output distributions are identical.*

Proof : Let us fix the function representing P and r (for both algorithms). We show by induction on the number of iterations that the output distribution of algorithms $\mathbb{A}''_r$ and $\mathbb{A}'_r$ are identical. We already proved for a fixed selection of points in iterations 1 to j that the next point chosen by algorithm $\mathbb{A}'_r$ is uniformly distributed among all points not selected so far. Clearly, the same is true for algorithm $\mathbb{A}''_r$ (if the distribution of the indices is uniform then so is the distribution of the corresponding points). Thus the induction step holds and thus the output distributions of both algorithms are identical. $\square$

We have already proven that $\mathbb{A}'$ is a property tester for property Π. Since the output distributions of $\mathbb{A}''$ and $\mathbb{A}'$ are identical for every input point set we have shown that algorithm $\mathbb{A}''$ is also a property tester. But $\mathbb{A}''$ uses a non-adaptive sampling. Finally, we show that there is an algorithm $\mathbb{A}'''$ that samples a set S of $q(\epsilon, n)$ points uniformly at random and decides based on S and its internal coin flips.

Algorithm $\mathbb{A}'''$ first samples a set S of $q(\epsilon, n)$ indices uniformly at random and then chooses a permutation $(i_1, \cdots, i_{q(\epsilon,n)})$ of S uniformly at random. It then runs algorithm $\mathbb{A}''$ and answers the j-th query of algorithm $\mathbb{A}''$ by returning the point stored at index i_j.

Claim 19 *Let* P *be any point set in* $\mathbb{R}^d$. *If algorithms* $\mathbb{A}'''$ *and* $\mathbb{A}''$ *are invoked with oracle access to the same function representing* P *then their output distributions are identical.*

Proof : As before, we fix the function representing point set P and the random choice of r. We show by induction on the number of iterations that both algorithms have the same output distribution. It suffices to show that the choice of the index in iteration $j+1$ is uniformly distributed among the indices not chosen so far. For every index $i \in [n] \setminus \{i_1, \ldots, i_j\}$ it holds that

$$\mathbf{Pr}[i_{j+1} = i \mid i_\ell \neq i \text{ for } 1 \leq \ell \leq j] \;=\; \frac{q(\epsilon,n)-j}{n-j} \cdot \frac{(q(\epsilon,n)-j-1)!}{(q(\epsilon,n)-j)!} \;=\; \frac{1}{n-j} \;.$$

Thus the index is chosen uniformly at random from the set of remaining indices. Hence, the output distributions of algorithms $\mathbb{A}'''$ and $\mathbb{A}''$ are identical. $\square$

Algorithm $\mathbb{A}'''$ satisfies the statement of Lemma 16 what completes the proof of Lemma 16. $\square$

5.2.2 Lower bound for testing for convex positions

Equipped with the framework of canonical property testers developed in Section 5.2.1 and in Lemma 16, we are ready now to prove our lower bound for testing for convex position.

Lemma 20 *Every property tester (with one sided error) for convex position has query complexity* $\Omega(\sqrt[d+1]{n^d/\epsilon})$.

Proof : The proof is by contradiction. Assume there is a property tester $\mathbb{A}$ for convex position with query complexity $q(\epsilon, n) = o(\sqrt[d+1]{n^d/\epsilon})$. We first want to prove that there is a canonical tester with the same query complexity:

Claim 21 *Let* $\mathbb{A}$ *be a property tester for convex position of point sets with query complexity* $q(\epsilon, n)$. *Then there is a property tester* $\mathbb{A}''$ *for convex position of point sets that samples a set* S *of* $q(\epsilon, n)$ *points uniformly at random and accepts if and only if the points in* S *are in convex position.*

Proof : By Lemma 16 we know that there is a property tester $\mathbb{A}'$ that samples a set of $q(\epsilon, n)$ points uniformly at random and decides whether to accept the input or to reject it on the base of S and its internal coin flips. Let $\mathbb{A}''$ be an algorithm that samples a set S of $q(\epsilon, n)$ points uniformly at random and that accepts if and only if S is in convex position. We want to prove that $\mathbb{A}''$ is a property tester. Clearly, algorithm $\mathbb{A}''$ accepts every point set in convex position. Thus we have to prove that it rejects every point set that is ϵ-far from convex position. We show this indirectly by proving that if algorithm $\mathbb{A}'$ rejects a sample set S then so does algorithm $\mathbb{A}''$. By definition a property tester has one-sided error and so $\mathbb{A}'$ must accept if the sample is in convex position. But if the sample S is not in convex position then $\mathbb{A}''$ rejects. It follows that $\mathbb{A}''$ always rejects when $\mathbb{A}'$ rejects. Since $\mathbb{A}'$ is a property tester it rejects every point set P that is ϵ-far from disjoint with probability at least $2/3$. Hence algorithm $\mathbb{A}''$ also rejects such a set P with probability at least $2/3$. Since $\mathbb{A}''$ accepts every point set in convex position we have shown that $\mathbb{A}''$ is a property tester for convex position. This proves Claim 21. □

Let us denote by $\mathbb{A}''$ the property tester as obtained in Claim 21. In the remainder of the proof we construct a point set P of n points that is ϵ-far from convex position and show that this point set is accepted by $\mathbb{A}''$ (observe that $\mathbb{A}''$ is uniquely defined by $q(\epsilon, n)$) with probability at least $1/2$. Since P is ϵ-far from convex position this is a contradiction to the fact that $\mathbb{A}''$ is a property tester (which would require that P is rejected with probability at least $2/3 > 1 - 1/2 = 1/2$).

We start with an overview of our construction and then present it in details. The idea is to construct a point set P that consists of $\epsilon n + 1$ sets W_i of size $d + 1$ (and some additional points). Each set W_i consists of a set F_i of d extreme points that form a facet of the convex hull and one point q_i that is located infinitesimally close to the facet but in the interior of the convex hull. The point set has the property that every subset $S \subseteq P$ of size more than $d + 1$ is not in convex position if and only if there is a set $W_i \subseteq S$.

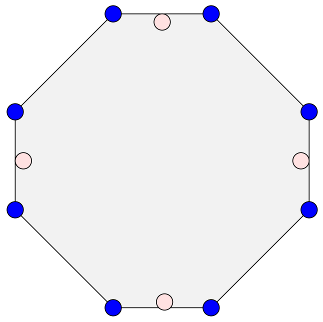

Figure 4: An example for the lower bound construction in 2D.

Now let S be a point set of size s chosen uniformly at random from P. We apply Lemma 3 with $p = 1/2$ and obtain that for

$$s \;\leq\; (n-d) \cdot \left(\frac{1}{2(\epsilon n + 1)}\right)^{1/(d+1)} \;=\; \Theta\left(\sqrt[d+1]{n^d/\epsilon}\right)$$

our sample set S does not contain one of the sets W_i with probability at least $1/2$. Hence S is in convex position with probability at least $1/2$ and thus algorithm $\mathbb{A}''$ is not a property tester, if $q(\epsilon, n) \leq (n-d) \cdot \left(\frac{1}{2(\epsilon n+1)}\right)^{1/(d+1)}$.

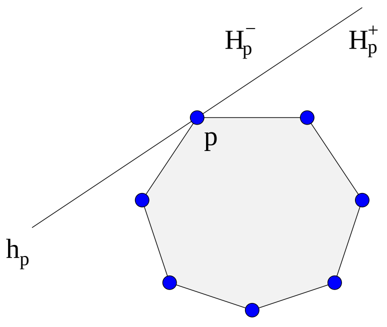

Figure 5: Step 1: We start with a point set in convex position.

Detailed construction. For a set X and a point p in $\mathbb{R}^d$ let $dist(X, p) = \min_{x \in X} ||x - p||$. Let P_1 be a set of $n - d \cdot (\epsilon n + 1)$ points in general and convex position and let $\epsilon < 1/d - 1/n$. Furthermore, let $P_2 = \{p_1, \ldots, p_k\} \subseteq P_1$ be a subset of size $k = \epsilon n + 1$. For each point $p \in P_1$ let h_p denote a hyperplane (a tangent to $\mathcal{CH}(P_1)$) such that $h_p \cap \mathcal{CH}(P_1) = p$. Next, let $\mathcal{H}_p^+, \mathcal{H}_p^-$ denote the closed halfspaces induced by h_p such that $\mathcal{H}_p^+$ denotes the halfspace containing P_1 (Figure 5). Let

$$\beta \;=\; \min_{p,q \in P_1, p \neq q} dist(h_p, q) \;.$$

Observation 22 *Every point set $\widetilde{P}_1$ obtained from P_1 by perturbing each of the points in P_1 by a distance of less than $\beta/2$ is in convex position.*

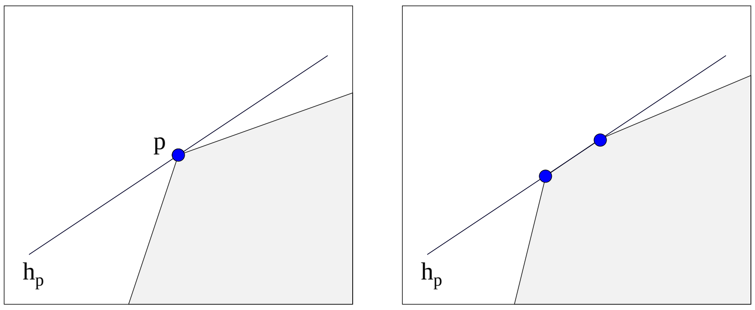

Figure 6: Step 2: We replace some points by a set F_i of $d = 2$ points.

Instead of perturbing the point set we replace each point $p_i \in P_2$ by a set F_i, $i \in [k]$, of d points (Figure 6). We do this replacement in such a way that we obtain a point set P_3 in general and convex position. For a point $p \in \mathbb{R}^d$, let $\mathbb{B}(p, \beta/3)$ denote the d-dimensional ball with center p and radius $\beta/3$. We choose each F_i such that

- $q \in h_p \cap \mathbb{B}(p, \beta/3)$ for each $q \in F_i$,
- $P_3 = P_1 \setminus P_2 \cup F$ is in general position, where $F = \bigcup_{i \in [k]} F_i$.

Clearly, we can choose the F_i such that P_3 is in general position because we can move the points within $\mathbb{B}(p_i, \beta/3) \cap h_{p_i}$ to destroy every degenerate cases. We observe that P_3 is in convex position since h_{p_i} is a witness for the fact that the points in F_i are extreme points in P_3.

We obtain the set P from P_3 by adding a set $Q = \{q_1, \dots, q_k\}$ of k points to P_3. Let $W_i = F_i \cup \{q_i\}$ for each $q_i \in Q$. We want to choose Q in such a way that:

- $P = P_3 \cup Q$ is ϵ-far from convex position, and
- if a subset $S \subseteq P$ contains no set W_i, $i \in [k]$, completely, then S is in convex position.

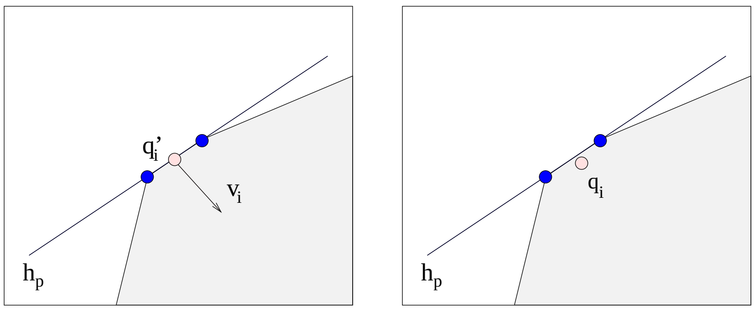

Figure 7: Step 3: We place another point in the interior of $\mathcal{CH}(F_i)$ and move it slightly along v_i into the interior of the convex hull.

We now give a construction that satisfies these properties. For each $i \in [k]$, let q'_i denote a point in the interior of the ($(d-1)$-dimensional) convex hull of F_i and let v_i denote a vector pointing from q'_i in the interior of $\mathcal{CH}(P_3)$ (Figure 7).

Let $q_i(\beta)$ denote the point $q'_i + \beta \cdot v_i$. For $i \in [k]$ and $p \in F_i$ let $h_{F_i,p}(\beta)$ denote the hyperplane induced by $F_i \setminus \{p\} \cup \{q_i(\beta)\}$. Furthermore, let $\mathcal{H}^+_{F_i,p}(\beta)$ and $\mathcal{H}^-_{F_i,p}(\beta)$ denote the halfspaces induced by $h_{F_i,p}(\beta)$ such that $\mathcal{H}^+_{F_i,p}(\beta)$ contains P for $\beta = 0$. Since P_3 is in general position there is a value β^* such that for every $i \in [k]$ and every $p \in F_i$ the halfspace $\mathcal{H}^+_{F_i,p}(\beta^*)$ contains $P_3 \setminus \{p\}$ and such that $P_3 \cup Q$ is in general position (where Q is defined as follows). We choose $q_i = q_i(\beta^*)$ and $Q = \bigcup_{i \in [k]} q_i$. We now have to prove that Q has the required properties:

Claim 23 *The set $P = P_3 \cup Q$ is ϵ-far from convex position. If a subset $S \subseteq P$ does not contain a set W_i completely, then S is in convex position.*

Proof : Assume P is ϵ-close to convex position. Then there is a set R of at most ϵn points such that $P \setminus R$ is in convex position. Since $|R| \leq \epsilon n$ there is a set W_i completely contained in $P \setminus R$. Furthermore, we know that by the size of P and R there must be at least one further point q in $P \setminus R$. By the choice of Q we know that q is in $\mathcal{H}^{+}_{F_i,p}(\beta^*)$ for each $p \in F_i$. Hence q_i is in the interior of $\mathcal{CH}(F_i \cup \{q\})$ which contradict the fact that $P \setminus R$ is in convex position.

It remains to prove the second statement. Let $S \subseteq P$ a subset that contains no set W_i completely. Without loss of generality, we assume that it contains exactly d points of each set W_i. If these points are the points in F_i then by our construction the hyperplane h_{p_i} is a witness that the points in $W_i = F_i$ are extreme. Otherwise, one of the points in F_i is not in S. Let is call the missing point p. Then $h_{F_i,p}(\beta^*)$ is a witness that the points are extreme points. Thus for every point $p \in S$ we have a witness that p is extreme. Hence S is in convex position. □

Next, we want to use our probabilistic Lemma 5 to give an upper bound on the probability that a sample set of size s chosen uniformly at random contains one of k sets W_i completely. We apply Lemma 5 with $p = 1/2$ and obtain that for

$$s \;\leq\; (n-d)\cdot\left(\frac{1}{2(\epsilon n+1)}\right)^{1/(d+1)} \;=\; \Theta\left(\sqrt[d+1]{n^f/\epsilon}\right)$$

a sample set S of size s chosen uniformly at random from P does not contain one of the W_i with probability at least $1/2$. Since $q(\epsilon, n) = o(\sqrt[d+1]{n^d/\epsilon})$, we have for sufficiently large n

$$q(\epsilon,n) \;\leq\; (n-d)\cdot\left(\frac{1}{2(\epsilon n+1)}\right)^{1/(d+1)} \quad .$$

It follows that there exists a point set P that is ϵ-far from convex position and that is accepted by algorithm $\mathbb{A}''$ with probability at least $1/2$. Since the existence of a property tester for convex position with query complexity $q(\epsilon, n)$ implies that algorithm $\mathbb{A}''$ is a property tester, it follows that there is no property tester with query complexity $o(\sqrt[d+1]{n^d/\epsilon})$. □

5.3 Complexity of testing convex position

The following theorem summarizes the results in this section, and it states both, the query complexity of the obtained tester and its running time complexity.

Theorem 24 *Let* P *be a point set of* n *points in the* $\mathbb{R}^d$. *Let* $T(n)$ *be the running time of the fastest known algorithm [5] to decide if a point set is in convex position; currently* $T(n) = \mathcal{O}(n \cdot \log^{\mathcal{O}(1)} h + (nh)^{\frac{\lfloor d/2 \rfloor}{\lfloor d/2 \rfloor + 1}} \cdot \log^{\mathcal{O}(1)} n)$, *with* h *denoting the number of extreme points of the set.*

Then there is a property tester for the convex position property with query complexity $\mathcal{O}(\sqrt[d+1]{n^d/\epsilon})$ *and running time* $\mathcal{O}(T(\sqrt[d+1]{n^d/\epsilon}))$. *Furthermore, every property tester for convex position has a query complexity of* $\Omega(\sqrt[d+1]{n^d/\epsilon})$.

Proof : Follows from Lemma 15 and Lemma 20. □

6 Delaunay triangulations

In this section we consider the problem of testing another classical structure in computational geometry which is the Delaunay triangulation (see, e.g., [13, 17]). For a given point set P in $\mathbb{R}^2$, the *Delaunay triangulation* is a collection of edges on P (which is a triangulation of P) such that for each edge we can find a circle containing the edge's endpoints but not containing any other points. Our goal is to develop a test to check whether a triangulation is Delaunay or it is far from the Delaunay triangulation. [5]

We assume the triangulation is given as a planar map. It is known that the Euclidean minimum spanning tree (EMST) for the input point set in $\mathbb{R}^2$ is a subgraph of the Delaunay triangulation, and in a companion paper [10] (see also [9]) we presented a property tester for the EMST. The distance measure we used was the *edit distance*: a graph is ϵ-far from EMST if one needs to delete or insert at least an ϵ fraction of the edges to obtain an EMST. Thus it seems natural to apply edit distance also to Delaunay triangulations. But it turns out that this measure makes the testing very hard, since a single vertex insertion might change the whole triangulation.

Theorem 25 *There is no sublinear property $\frac{1}{4}$-tester for Delaunay triangulations using edit distance (edge deletion and insertion). This result holds even for property testers with two-sided error.*

Proof : We will construct two distributions D_1, D_2 over geometric graphs with vertices in $\mathbb{R}^2$. Graphs from D_1 will be proper Delaunay triangulations and graphs from D_2 will be ϵ-far from a Delaunay triangulation in the edit distance and for some small enough constant ϵ.

Distribution D_1 will only contain a single instance, which is constructed as follows. The vertices have the following coordinates: $(1,0), (1, \frac{1}{n}), (1, \frac{2}{n}), \ldots, (1, \frac{\lceil n/2 \rceil}{n}), \ldots, (0, \frac{1}{2n}), (0, \frac{3}{2n}), (0, \frac{5}{2n}), \ldots$. The graph G in D_1 is the Delaunay triangulation of these points. Distribution D_2 is obtained by choosing one vertex from G uniformly at random, deleting this vertex and re-triangulating the created hole. This way, we obtain a Delaunay triangulation G' over $n-1$ points. Now we consider the triangles from G that are also in G' and select one of them uniformly at random. We place a new point ν at the center of gravity of this triangle. It can be easily seen that there is a constant $\alpha > 0$, such that any such point lies in the circumcircle of αn triangles of G'. We connect ν with the vertices of the triangle it is contained in. Thus, we have obtained a triangulation that is ϵ-far from Delaunay for $\epsilon < \alpha$.

It remains to show that no algorithm with $o(n)$ query complexity can distinguish between these two distributions with probability at least 2/3. Let us consider an arbitrary property tester $\mathcal{A}$ with query complexity $q(n)$ and let us fix the random bits of $\mathcal{A}$. We first consider the result of running $\mathcal{A}$ on G. Let us mark every vertex that accessed by the algorithm either by querying its position or querying an incident edge. Now let us consider a random graph from G'. Clearly, G' can be distinguished from G only if one of the marked vertices are changed during the construction of G'. But in this construction only a constant number of vertices is changed and so the probability that a marked vertex is changed is $O(q(n)/n)$. Summing up over all strings of random bits we obtain that the probability of accepting G' is at least $2/3 - O(q(n)/n)$, since $\mathcal{A}$ must accept G with probability at least 2/3. Since $\mathcal{A}$ is a property tester, it must reject G' with probability at least 2/3, which implies that $q(n) = \Omega(n)$. □

Testing Delaunay triangulation using different distance measures. We have seen that the Delaunay triangulation is less suited for property testing than the EMST. However, there is a fast property test for the Delaunay triangulation in another distance measure. For a given triangulation $\mathcal{T}$, a *flip* of an edge e

[5]Here we sketch an interesting application of such a test during the (exact) computation of the Delaunay triangulation with an incremental algorithm: We compute the Delaunay triangulation using floating point arithmetic and fix the triangulation at the end of the algorithm (e.g., with edge flips now using exact arithmetic). To guarantee that we keep close to the correct triangulation after $\mathcal{O}(1)$ vertex insertions we check whether the current triangulation is close to the Delaunay triangulation. If our test algorithm rejects, we fix the triangulation and continue. This way we can avoid that small errors at the beginning of the computation lead to large structural defects in the final triangulation.

is a triangulation obtained from $\mathcal{T}$ by removal of edge e and inserting another edge $e^* \neq e$ to the square obtained after the removal.

Definition 26 *A triangulation is ϵ-far from being Delaunay if there are at least $\epsilon\, n$ edges that can be flipped to improve the minimal angle locally.*

Since any triangulation contains $\Theta(n)$ "flippable" edges [19] the above definition gives us a weak but still useful distance measure for the Delaunay property. One can easily show that sampling and checking $\mathcal{O}(1/\epsilon)$ edges suffices to test whether a triangulation is ϵ-far from Delaunay.

Theorem 27 *If we define a triangulation to be ϵ-far from being Delaunay if there are at least $\epsilon\, n$ edges that can be flipped to improve the minimal angle locally, then there exists an ϵ-tester with the running time of $\mathcal{O}(1/\epsilon)$.* □

7 Deterministic lower bounds

In this section we briefly discuss that if one requires only deterministic algorithms, for most of geometric properties $\Omega(n)$ time is required.

Our model for the deterministic algorithms is as follows. The input is given as an array of input items. The array has size n and we may access any item by its index in constant time. Let A be a deterministic $\frac{1}{2}$-tester for property Q. Let I be an instance of size n with property Q. By the definition, A must accept I. Let $T_A(n)$ be the running time of A. Clearly, A can access at most $T_A(n)$ items of I. We color the accessed items red and all other items blue.

Since A is deterministic, changing the blue items does not affect the outcome of the algorithm. To obtain a lower bound of $\Omega(n)$, it is enough to show that if $T_A(n) < \frac{1}{2}n$ and A examines only red items, then we can construct from I an instance that is $\frac{1}{2}$-far from P by changing only blue items regardless how the red items are chosen by A. Therefore, we can easily conclude with the following theorem.

Theorem 28 *There is no deterministic $\frac{1}{2}$-tester with $o(n)$ query complexity for the following problems:*

Property	*Structure*
Sorting	*Array*
Convex Position	*Point Set*
Disjointness of Polytopes	*Point Set*
Disjointness of Objects	*Object Set*
Euclidean Minimum Spanning Tree	*Graph*
Delaunay Triangulation	*Planar Map*

□

8 Final comments

A preliminary version of this paper appeared in a form of an extended abstract as a part of [9]. The conference version [9] has been then split into two parts, one which forms the current paper and another which appears in [10]. [10] contains a property testing algorithm for testing if a given graph is an Euclidean spanning tree of a given set of points in $\mathbb{R}^2$. Since that property testing algorithm is of different flavor and is quite complex, we decided to split the conference version of the paper into two journal publications.

Acknowledgements

We thank Stefan Funke and Emo Welzl for their helpful comments on this work.

References

[1] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. *SIAM Journal of Discrete Mathematics*, 16(3): 393–417, 2003.

[2] H. Alt, R. Fleischer, M. Kaufmann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig. Approximate motion planning and the complexity of the boundary of the union of simple geometric figures. *Algorithmica*, 8: 391–406, 1992.

[3] Z. Bar-Yossef, Ravi Kumar, and D. Sivakumar. Sampling algorithms: Lower bounds and applications. In *Proceedings of the 33th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 266–275, 2001.

[4] C. Carathéodory. Über den Variabilitätsbereich der Fourierschen Konstanten von positiven harmonischen Funktionen. *Rendiconto del Circolo Matematico di Palermo*, 32: 193–217, 1911.

[5] T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete & Computational Geometry*, 16(3): 369–387, 1996.

[6] D. Cohen-Or, Y. Chrysanthou, C. T. Silva, and F. Durand. A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization & Computer Graphics*, 9(3):412–431, 2003.

[7] A. Czumaj and C. Sohler. Property testing with geometric queries. In *Proceedings of the 9th Annual European Symposium on Algorithms (ESA)*, pp. 266–277, 2001.

[8] A. Czumaj and C. Sohler. Abstract combinatorial programs and efficient property testers. *SIAM Journal on Computing*, 34(3): 580–615, 2005.

[9] A. Czumaj, C. Sohler, and M. Ziegler. Property testing in computational geometry. In *Proceedings of the 8th Annual European Symposium on Algorithms (ESA)*, pp. 155–166, 2000.

[10] A. Czumaj, C. Sohler, and M. Ziegler. Testing Euclidean minimum spanning trees in the plane. Accepted to *ACM Transactions on Algorithms*, 2007.

[11] F. Ergün, S. Kannan, S. Ravi Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60: 717–751, 2000.

[12] E. Fischer. The art of uninformed decisions. a primer to property testing. *Bulletin of the European Association for Theoretical Computer Science*, 75: 97–126, 2001.

[13] S. Fortune. Voronoi diagrams and Delaunay triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 20, pages 377–388, CRC Press, Boca Raton, FL, 1997.

[14] O. Goldreich. Combinatorial property testing (a survey). In P. Pardalos, S. Rajasekaran, and J. Rolim, eds., *Proceedings of the DIMACS Workshop on Randomization Methods in Algorithm Design*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 45–59, 1997. American Mathematical Society, 1999.

[15] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4): 653–750, 1998.

[16] O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. *Random Structures and Algorithms*, 23(1): 23–57, 2003.

[17] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(4): 381–413, 1992.

[18] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2: 127–151, 1987.

[19] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3): 333–346, October 1999.

[20] K. Mehlhorn, S. Näher, M. Seel, R. Seidel, T. Schilz, S. Schirra, and C. Uhrig. Checking geometric programs or verification of geometric structures. *Computational Geometry: Theory and Applications*, 12: 85–103, 1999.

[21] K. Mehlhorn, T. C. Shermer, and C. K. Yap. A complete roundness classification procedure. In *Proceedings of the 13th Annual ACM Symposium on Computational Geometry (SoCG)*, pp. 129–138, 1997.

[22] D. Ron. Property testing. In P. M. Pardalos, S. Rajasekaran, J. Reif, and J. D. P. Rolim, eds., *Handobook of Randomized Algorithms*, volume II, pp. 597–649. Kluwer, 2001.

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