Alcom-FT Technical Report Series
ALCOMFT-TR-02-44

Property Testing with Geometric Queries

(Extended abstract)

Artur Czumaj[*] and **Christian Sohler**[†]

May 7, 2002

Abstract

This paper investigates geometric problems in the context of property testing algorithms. Property testing is an emerging area in computer science in which one is aiming at verifying whether a given object has a predetermined property or is "far" from any object having the property. Although there has been some research previously done in testing geometric properties, prior works have been mostly dealing with the study of *combinatorial* notion of the distance defining whether an object is "far" or it is "close"; very little research has been done for *geometric* notion of distance measures, that is, distance measures that are based on the geometry underlying input objects.

The main objective of this work is to develop sound models to study geometric problems in the context of property testing. Comparing to the previous work in property testing, there are two novel aspects developed in this paper: geometric measures of being close to an object having the predetermined property, and the use of geometric data structures as basic primitives to design the testers. We believe that the second aspect is of special importance in the context of property testing and that the use of specialized data structures as basic primitives in the testers can be applied to other important problems in this area.

We shall discuss a number of models that in our opinion fit best geometric problems and apply them to study geometric properties for three very fundamental and representative problems in the area: testing convex position, testing map labeling, and testing clusterability.

1 Introduction

A classical problem in computer science is to verify if a given object possesses a certain property. For example, we want to determine if a boolean formula is satisfiable, or if a set of polygons in the Euclidean plane is intersection free. In its very standard formulation, the goal is to give an exact solution to the problem, that is, to provide an algorithm that always returns a correct answer. In many situation, however, this formulation is too restrictive, for example, because there is no fast (or just fast enough) algorithm that gives the exact solution. Very recently, many researchers started studying a relaxation of the "exact decision task" and consider various forms of approximation algorithms for decision problems. In *property testing* (see, e.g., [3, 4, 7, 10, 12, 13, 14, 15, 18]), one considers the following class of problems:

[*]Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ 07102, USA. Email: czumaj@cis.njit.edu.

[†]Heinz Nixdorf Institute and Department of Mathematics & Computer Science, University of Paderborn, D-33095 Paderborn, Germany. Email: csohler@uni-paderborn.de. Research supported in part by an SBR grant No. 421090, by DFG grant Me872/7-1, and by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

Let $\mathfrak{C}$ be a class of objects, $\mathfrak{O}$ be an unknown object from $\mathfrak{C}$, and $\mathcal{Q}$ be a fixed property of objects from $\mathfrak{C}$. The goal is to determine (possibly probabilistically) if $\mathfrak{O}$ has property $\mathcal{Q}$ or if it is *far* from any object in $\mathfrak{C}$ which has property $\mathcal{Q}$, where distance between two objects is measured with respect to some distribution $\mathcal{D}$ on $\mathfrak{C}$.

The motivation behind this notion of property testing, is that while relaxing the exact decision task, we expect the testing algorithm to be significantly more efficient than any exact decision algorithm, and in many cases, we achieve this goal by exploring only a small part of the input. And so, for example, in [4] it is shown that all first order graph properties of the type "$\exists\forall$" can be tested in time independent of the input size (see also, [13, 14, 18] for some other most striking results).

In the standard context of property testing, the first general study of geometric properties appeared in [7]. In this paper the authors studied property testing for classical geometric problems like being in convex position, for disjointness of geometric objects, for Euclidean minimum spanning tree, etc. Roughly at the same time, in [3], property testing for some clustering problems has been investigated. In [10], the problem of testing if a given list of points in $\mathbb{R}^2$ represents a convex polygon is investigated. In all these papers, the common measure of being close to having the predetermined property was the *Hamming distance*. That is, for an object $\mathfrak{O}$ from a class $\mathfrak{C}$, a property $\mathcal{Q}$, and a real ε, $0 \leq \varepsilon \leq 1$, we say $\mathfrak{O}$ is ε-far from having property $\mathcal{Q}$, if any object $\mathfrak{O}$ from $\mathfrak{C}$ that has property $\mathcal{Q}$ has the Hamming distance at least $\varepsilon \cdot |\mathfrak{O}|$ from $\mathfrak{O}$. The Hamming distance is a standard measure to analyze combinatorial problems, but in the opinion of the authors, other more geometric distance measures should also be considered in the context of Computational Geometry. The reason is that this measure does not explore geometry underlying investigated problems, but only their combinatorial structure (how many "atom" objects must be modified to transform the object into one possessing the requiring property). This issue has been partly explored in the context of the *metrology of geometric tolerancing* [5, 6, 8, 17, 19, 20]. In this area (motivated by manufacturing processes) one considers the problems of verifying if a geometric object is within some given tolerance from having certain property $\mathfrak{Q}$. In geometric tolerancing, the researchers have been studying among others, the "roundness property," the "flatness property," etc. [5, 6, 8, 17]. We emphasize, however, that there is a major difference between the notion of geometric property testing and geometric tolerancing in that in the former one allows to reject (as well as accept) any object that does not satisfy the property $\mathfrak{Q}$, while in geometric tolerancing one should accept such an object if it is within given tolerance.

Our contribution. This paper is partly of a methodological character. The main objective of this paper is to develop proper models to study geometric problems in the context of property testing. We shall discuss a number of models that in our opinion best fit geometric problems and apply them to study geometric properties for the most fundamental problems in the area. Comparing to the previous work in property testing, in the current paper we develop two main novel ideas:

- ☞ geometric measures of being close to an object having the predetermined property, and
- ☞ the use of geometric data structures to develop the testers.

We discuss these two issues in details in Sections 2 and 3 while testing *convex position*. We demonstrate the need of geometric distance measures for geometric problems and propose three new models that in our opinion suit best to study geometric properties. We show also that the complexity measures used in standard property testing have to be modified in order to achieve something non-trivial using geometric distance measures. We propose a model of computation that uses *queries for geometric primitives* (in this case range queries) as its basis and discuss its use and practical justifications. Finally, we illustrate all these issues by designing property testing algorithms for convex position. Unlike in the model investigated in [7], our testing algorithms run in time either completely independent of the input size or only with a polylogarithmic dependency, and we believe that they fit much better the geometry underlying the problem of testing convex position.

In Section 4, we investigate the *map labeling* problem. We first show that in the classical property testing setting (that uses uniform sampling of the input points) this problem does not have fast testing algorithms. Next, we show that by using geometric queries as basic operations one can obtain very efficient testing algorithms. We present an ε-tester for map labeling that requires only $\mathrm{poly}(1/\varepsilon)$ range queries of the form: *"What is the i-th point in the orthogonal range R ?"*

Then, in Section 5, we consider *clustering problems* in our context and provide efficient testers for clustering problems in most reasonable geometric models. The goal of a clustering problem is to partition a point set in $\mathbb{R}^d$ into k different clusters such that the cost of each cluster is at most b. We consider three different variants of the clustering problem (see, e.g., [3]): *radius clustering*, *discrete radius clustering*, and *diameter clustering*. We say that a set of points is *ε-far from clusterable* with k clusters of size b, if there is no clustering into k clusters of size $(1+\varepsilon)\,b$. We show that it is possible to test clusterability using $\mathcal{O}(k/\varepsilon^d)$ oracle range queries.

Comparing our results to those in [3], we use a more powerful oracle but we also have a more restrictive distance measure. Using our distance measure and the classical oracle from [3], it is impossible to design a sublinear property tester for this problem.

Further, we show how to use our tester to maintain (under insertion and deletion of points) an approximate k-clustering in $\mathbb{R}^d$ of size at most $(1+\varepsilon)$ times the optimum in time $\mathrm{polylog}(n)$ for any constants k, d, and ε. Here, n denotes the current number of points.

2 Testing convex position

Let us first consider the classical problem of testing if a point set P in the plane is in *convex position* (that is, the interior of the convex hull $conv(P)$ contains no point from P, or equivalently, all points in P are *extreme*). Our goal is to consider a practical situation in which we allow some relaxation of the exact decision test and we consider the following type of testers:

☞ If P is in convex position, then the tester must accept the input.

☞ If P is "far" from convex position, then the tester "typically" rejects the input.

☞ If P is not in convex position, but it is close to being so, then the answer may be arbitrary.

In order to use this concept we must formalize some of the notions used above. First of all, we assume a tester is a possibly randomized algorithm and, following standard literature in this area, by "typically" we shall mean that the required answer is output with probability at least $\frac{2}{3}$, where the probability is over the random choices made by the tester (and thus, this lower bound of $\frac{2}{3}$ is independent of the input).

2.1 Distance measures — far or close

A more subtle issue is what do we mean by saying that P is "far" from convex position. We pick a parameter ε, $0 \le \varepsilon \le 1$, which will measure the quality of how "close" is P to convex position In the standard terminology used in the property testing literature (see, e.g., [13, 18]), one uses the following definition:

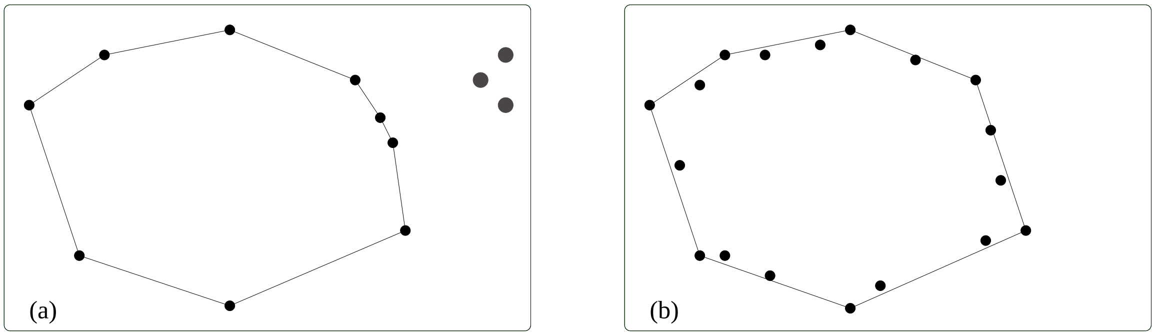

Figure 1: Which of the two point sets is "more convex?" In Figure (a) it is enough to delete only 3 points (those in the top right corner) to obtain a point set in convex position; in Figure (b) one has to remove much more points to do so. On the other hand, the points from Figure (a) are visually far from convex position, while points in Figure (b) look similar as they were in convex position; it is enough to perturbate them very little to obtain a point set in convex position.

Definition 2.1 **(Hamming distance)** *A point set* P *in the Euclidean space* $\mathbb{R}^d$ *is* ε-far *from convex position (according to the Hamming distance), if for any subset* $S \subseteq P$, $|S| \le \varepsilon \cdot |P|$, *set* $P \setminus S$ *is not in convex position.*

We found, however, that this measure often does not correspond to notions of the distance used in geometry (see, e.g., Figure 1). It tells only about combinatorial properties of the object at hand, but it tells very little about geometry behind the object. For example, do we want to accept an n-point set P if it contains $\frac{1}{2}\,\varepsilon\, n$ points that are very far away from the remaining points that are in convex position (as, for example, in Figure 1 (a))? Or perhaps, we consider such a set P as far from convex position? On the other hand, if P contains an ε fraction of points which make P non-convex, but after a very small perturbation of these points, the obtained set will be in convex position (see, e.g., Figure 1 (b)). Do we want to call such a point set ε-far from convex position or not?

It is clear that the distance notion is very application dependent, and in this paper we investigate various distance measures which should be of practical interest, and study basic problems from computational geometry for these distance measures.

We begin with a distance that measures how much the input points are allowed to be moved (perturbated) in order to transform them into being in convex position.

Definition 2.2 **(Perturbation measure)** *A point set* P *in the* d*-dimensional unit cube is* ε-far *from convex position (according to the Perturbation measure), if for any perturbation of points in* P *that moves any point by distance at most* ε*, the resulting point set is not in convex position.*

Because of scaling, the fact that P is enclosed by the unit cube is assumed without loss of generality.

We introduce also another measure that although very similar to the Perturbation measure, will be more useful for our applications.

Definition 2.3 **(Neighborhood measure)** *A point set* P *in the* d*-dimensional unit cube is* ε-far *from convex position (according to the Neighborhood measure), if there exists a point* $p \in P$ *for which the* d*-dimensional ball of radius* ε *with center at* p *does not intersect the boundary of* $\mathrm{conv}(P)$.

The next measure is more related to the volume discrepancy of the convex hull of the input points. It differs significantly from the Perturbation and Neighborhood measures, because this measure is relative to the volume of $conv(P)$.

If a point set P is in convex position, then all points in P lie on the boundary of convex hull $conv(P)$ and therefore $conv(P)$ is also the maximal convex hull defined by any (non-trivial) subset of P whose interior contains no point from P. In view of this, we may want to consider P to be close to convex position, if a maximum (with respect to the volume) convex hull defined by a subset of P that contains no point from P in its interior[1] is almost the same as $conv(P)$. If we use the volume measures for these two objects, then we get the following definition:

Definition 2.4 **(Volume measure)** *A point set P in $\mathbb{R}^d$ is* ε-far *from convex position (according to the Volume measure), if $\frac{vol(EmpInt(P))}{vol(conv(P))} \leq 1 - \varepsilon$, where $vol(X)$ denotes the volume of object X and $EmpInt(P)$ is a maximum volume convex hull defined by a subset of P that contains no point from P in its interior.*

Now, we are ready to formally define property testing algorithms.

Definition 2.5 **(ε-Testers)** *An algorithm is called an* ε-tester *for a property $\mathcal{Q}$, if it always accepts any input satisfying property $\mathcal{Q}$ and with probability at least $\frac{2}{3}$, rejects any input that is ε-far from satisfying property $\mathcal{Q}$.*

Throughout the paper, we say P is ε-*close* to convex position if it is not ε-far from convex position.

2.1.1 Relations between different measures of closeness.

How can we relate the four measures defined in Definitions 2.1–2.4? As we observed above, a point set P can be close to convex position according to the Hamming distance, even if it is far (very far!) from convex position according to the Perturbation, the Neighborhood, and the Volume measures. Similarly, the opposite is also true: P may be very close to convex position according to the Perturbation, the Neighborhood, and the Volume measures even it it fails for the Hamming distance. But how about any relationship between the Perturbation, the Neighborhood, and the Volume measures?

The first lemma shows that the first two measures are somehow equivalent for asymptotic complexity of the testers. It holds for any cost measure of query complexity, because exactly the same tester is to be used.

Lemma 2.1 *There is an ε-tester for convex position according to the Perturbation measure with query complexity $T(n, \varepsilon)$ if and only if there is an $\Theta(\varepsilon)$-tester for convex position according to the Neighborhood measure with query complexity $T(n, \Theta(\varepsilon))$.* $\Box$

Unfortunately, we were unable to provide a similar relationship between the Neighborhood and the Volume measures. It seems to us that the latter one is more complicated. We can only prove some partial results about similarity of these two measures, for example:

Lemma 2.2 *An ε^d-tester for the Volume measure is an $\mathcal{O}(\varepsilon)$-tester for the Neighborhood measure.* $\Box$

[1] Observe that in general that may be many such maximum convex hulls.

3 A New Model using Geometric Queries

In the previous works on property testing, the complexity of a tester has been typically measured as the number of input "atom objects" inspected, that is, as the number of *queries* to the input. The form of the queries allowed for the algorithm depended on the input representation. And so, for example, if an input consists of a set of n points (as it is the case for testing if the points are in convex position), then it has been typically assumed that one can use queries of the form: "*what is the position of the* k*th point in the input*." In the standard query complexity additional computational work is not counted (for example, if we know positions of points in S, $S \subseteq P$, then the cost of computing a convex hull of S is not counted in the query complexity[2]). Our main observation is that this notion of query complexity often does not suit well to study geometric properties, or actually, to distance measures different than the Hamming distance. Indeed, if we want to check if a point set P is ε-close to convex position according to the Perturbation measure, then even a single point might be far away from the remaining points to make this property false. Therefore, the algorithm must find this point with probability at least $\frac{2}{3}$, and this clearly requires $\Theta(n)$ query complexity. Similar phenomenon holds also for the Neighborhood measure and the Volume measure.

This observation shows that in order to model property testing for geometric properties and in order to obtain very efficient (sublinear-time) algorithms one has to reconsider and change the notion of query complexity. Unlike in the very standard model, here we want to allow more complex queries: those using certain geometric properties of the input.

In most of geometric models (and in applications) even if the input is represented by positions of the points (or other geometric objects), very often one maintains some additional data structures for efficient and structured access to the input. One of the most fundamental abstract data structure maintained by many algorithms working with points are data structures for efficient answering *range queries* (cf. [1]). For the purposes of this paper we adopt a model of computation in which the basis operation is a range query to the input, and the *query complexity* is the number of range queries to the input.

[3]Formally, we are given an unknown set P of n points in $\mathbb{R}^d$ that is defined by an unknown function $\mathcal{F}_P : \mathbb{R}^d \times \mathbb{N} \to \mathbb{R}^d \cup \{\mathtt{empty}\}$ such that $\mathcal{F}_P(R, i)$ returns the ith point in a query range R (according to some unknown fixed order) or the symbol $\mathtt{empty}$, if there are less than i points in the query range R.

The model defined above uses a very powerful oracle since we are allowed to specify an arbitrary range when we query the oracle. To make our consideration of practical value, it seems reasonable to require that such an oracle must be efficiently implemented. Therefore, in this model we will restrict ourselves to the case that R is a (possibly unbounded) simplex. Most of the results presented in this paper hold even for *orthogonal range queries*. Such queries are supported by many well known data structures such as partition trees and cutting trees, as well as practical structures based on quad-trees or R-trees (see, e.g., [1] for a more detailed discussion). There are efficient data structures (see, e.g., [1]) to support our queries and a single query to such a data structure is usually performed very fast (i.e., much faster than processing the whole point set). We believe that the use of such range queries

---

[2]For example, in [7], it is shown that the query complexity for testing (according to Hamming distance) if a point set in $\mathbb{R}^d$ is in convex position is $\Theta(n^d/\varepsilon)^{1/(d+1)}$, while for $d \geq 4$, the "running time complexity" (which measures also the time required for all computations used by the tester) is $\mathcal{O}(n\,\mathrm{polylog}(1/\varepsilon) + (n/\varepsilon)^{(\lceil d/2\rceil)/(1+\lceil d/2\rceil)}\mathrm{polylog}(n))$, and it is quite possible that it is optimal. Thus, in the most basic case, for $d = 4$ and constant ε^{-1}, the query complexity is $\Theta(n^{3/4})$ while the "running time complexity" is $\mathcal{O}(n)$. This difference vanishes for $d = 2, 3$, because in this case very efficient (almost linear-time) algorithms for testing convex position are available.

[3]We formalize our model of computation only to inputs that are in the form of point sets; for other input types the model can be defined accordingly in a similar way.

is very natural in our context, since many applications (such as GISs) use data structures for range queries (e.g., R-trees) to answer other kinds of queries anyway.

In a similar way, depending on the problem at hand, one could assume that some other very basic *geometric queries* are available; we do not discuss this issue in more details however.

3.1 Property testing algorithms for convex position in the new model

In this section we present our first ε-tester for convex position. It works for the Neighborhood measure, and it shows that the use of geometric queries (orthogonal range queries) allows to beat the lower bounds discussed in Section 3 and obtain the query complexity of $\mathrm{polylog}(1/\varepsilon)$.

The input for our tester consists of a point set P in the d-dimensional unit cube and a real number ε, $0 < \varepsilon < 1$, which defines the quality of the tester.

```
CONVEXITY-TEST I (P, ε):
  S = ∅
  partition the unit cube into (2 √d/ε)^d sub-cubes of side-length ε/(2 √d)
  for each such sub-cube c do
      if c contains a point from P then add any such a point to S
  if S is in convex position then accept
                              else reject
```

In algorithm CONVEXITY-TEST I (P, ε) the operation of verifying if c contains a point from P as well as the operation of returning a point from $P \cap c$ is performed using orthogonal range queries.

Theorem 1 *Let P be a point set in the d-dimensional unit cube and let ε be a real number, $0 < \varepsilon < 1$. Algorithm* CONVEXITY-TEST I (P, ε) *is a property tester that accepts P only if P is ε-close (according to the Neighborhood measure) to convex position. It uses $\mathcal{O}((\sqrt{d}/\varepsilon)^d)$ orthogonal range queries.* □

Actually, we can slightly improve the complexity of CONVEXITY-TEST I (P, ε) and design an ε-tester that uses only $\mathcal{O}((\sqrt{d}/\varepsilon)^{d-1})$ orthogonal range queries.

In the previous sections we discussed testing convexity properties in geometric setting. Now, we give a tester for testing convexity properties of planar point sets using a distance measure that is related to the Hamming distance (in fact, the distance measure below is stronger). A tester for Hamming distance is presented in [7]. The main difference in our approach here is the use of geometric queries that leads a to substantial speed up.

Definition 3.1 *A set P of n points in the plane is* ε-far from being in convex position*, if at least εn points in P are not extreme. (A point is extreme if it belongs to the boundary of the convex hull of P.)*

It immediately follows.

Lemma 3.1 *A tester for the distance measure in Definition 3.1 is also a tester for the Hamming distance.* □

We can also prove the following lemma.

Lemma 3.2 *In the standard property testing model (see, e.g., [7, 13, 18]), there is no testing algorithm for the distance measure from Definition 3.1 that has $o(n)$ query complexity.* □

We can prove that the use of appropriate data structures for the geometric queries allows us to design a tester with logarithmic query complexity. We assume the input point set P is in general position.

Theorem 2 *There is a tester for convex position in the plane with query complexity $\mathcal{O}(\log n/\varepsilon)$ that uses only triangular range queries.* □

4 Map Labeling

In this section we consider the following basic *map labeling* problem:

Let P be a set of n points in the plane. Decide whether it is possible to place n axis-parallel unit squares such that

- *all squares are pairwise disjoint (labels do not overlap),*
- *each point is a corner of exactly one square (each point is labeled), and*
- *each square has exactly one point on its corners (each point has a unique label).*

If a set $\mathcal{S}$ of n squares satisfies the conditions above, then $\mathcal{S}$ is called a *valid labeling for* P. The map labeling problem is known to be $\mathcal{NP}$-complete and the corresponding optimization problem is known to have no approximation algorithm with ratio better than 2, unless $\mathcal{P} = \mathcal{NP}$ [11].

In this section we develop a property tester for the map labeling problem. We use the following Hamming distance measure:

Definition 4.1 *A set P of n points in the plane is ε-far from having a valid labeling, if we have to delete at least εn points to obtain a set of points that has a valid labeling.*

When we consider the standard property testing model [13, 18] that allows only to sample random subsets of P with a uniform distribution we can prove the following result.

Theorem 3 *For any constant δ, $0 < \delta < 1$, there is a positive constant ε such that there is no ε-tester for the labeling problem with $o(n^{1-\frac{1}{2\delta+1}})$ query complexity in the standard testing model.* □

We show now that if we use the computational model that allows/supports geometric queries we can design a tester with $\mathcal{O}(1/\varepsilon^3)$ query complexity. It is based on the approach developed in [9] and [16].

```
LABELTEST(P):
    choose a sample set 𝒮 of size 𝒪(1/ε) uniformly at random from P
    for each p ∈ 𝒮 do
        i = 0, 𝒯 = ∅
        Let S be the axis parallel square with center p and side length 16⌈1/ε⌉
        while i ≤ (16⌈1/ε⌉ + 2)² do
            Let q be the i-th point in the query range S
            if q ≠ ∅ then 𝒯 = 𝒯 ∪ {q}
        if 𝒯 does not have a valid labeling then reject
    accept
```

Theorem 4 *Algorithm* LABELTEST *is a tester for the labeling problem that has query complexity $\mathcal{O}(1/\varepsilon^3)$ and running time* $\exp(\mathcal{O}(1/\varepsilon^2)$. □

5 Clustering Problems

In this section we design testing algorithms for three geometric clustering problems. The goal of a clustering problem is to decide whether a set of n points P in $\mathbb{R}^d$ can be partitioned into k subsets (called *clusters*) $S_1, \ldots, S_k$ such that the *cost* of each cluster is at most b. There are several different ways to define the cost of a cluster. Let S be a set of points in $\mathbb{R}^d$. We consider the following variants:

Radius Clustering: The cost $\mathrm{cost}_R(S)$ of a cluster S is twice the minimum radius of a ball containing all points of the cluster.

Discrete Radius Clustering: The cost $\mathrm{cost}_{DR}(S)$ of a cluster S is the minimum radius of a ball containing all points of the cluster and having its center among the points from P.

Diameter Clustering: The cost $\mathrm{cost}_D(S)$ of a cluster S is the maximum distance between a pair of points of the cluster.

The goal of our property tester is to accept all instances that admit a clustering into k subsets of cost b and to reject with high probability those instances that cannot be clustered into k subsets of cost $(1+\varepsilon)\, b$.

Definition 5.1 *A point set P is (b,k)-clusterable for a cost measure $\mathrm{cost}()$, if there is a partition of P into sets $S_1, \ldots S_k$ such that $\mathrm{cost}(S_i) \le b$ for all $1 \le i \le k$. A point set P is ε-far from being (b,k)-clusterable, if for any partition of P into sets $S_1, \ldots S_k$ at least one set S_i has cost larger than $(1+\varepsilon)\, b$.*

In the standard context of property testing, Hamming distance (that is, a point set is ε-far from clusterable, if we have to remove at least εn points to make it clusterable) has been used before [3]. For the diameter clustering problem the distance measure used in [3] has the additional relaxation that a point set is ε-far from (b,k)-clusterable, if one has to remove εn points to make the set $((1+\varepsilon)\, b, k)$-clusterable. Thus, this definition assumes a geometric and a combinatorial relaxation of the corresponding decision problem. We require only the geometric relaxation.

Let us assume, without loss of generality, that $b = 1$ and thus we want to design a tester for the problem whether a point set P is $(1,k)$-clusterable for the three cost measures above. We partition $\mathbb{R}^d$ into grid cells of side length $\varepsilon/(3\sqrt{d})$. For each cell containing an input point, we choose arbitrary input point from the cell as its representative. Then, we compute whether the set of representatives is $(1,k)$-clusterable. If it is so, then we accept it, if it is not so, then we reject it. Clearly, any set of points that is $(1,k)$-clusterable is accepted by the algorithm. On the other hand, any instance that is ε-far from $(1,k)$-clusterable will be rejected. (This approach has been introduced in [2] to obtain a $(1+\varepsilon)$-approximation algorithm for the radius clustering problem.)

Our algorithms starts with an empty box with endpoints at infinity. Then we query for a point in this box. We allocate the corresponding grid cell and partition the box into the 3^d sub-boxes induced by the hyperplanes bounding the grid cell. Then we continue with one of these sub-boxes. If we find an empty sub-box, it will be marked. If there are only marked boxes the algorithm terminates.

So far, our partition into grid cells works fine, if there are many points in a single cell. On the other hand, if no two points are in the same grid cell, the algorithm has $\Omega(n)$ query complexity. Thus we need an upper bound on the number of grid cells, whose representative may form a cluster.

Lemma 5.1 *Let S be a set of points in $\mathbb{R}^d$ no two points of which belonging to the same cell of a grid of size $\varepsilon/(3\sqrt{d}) < 1$. If $\mathrm{cost}(S) \le 1$ for any of the three cost measures described above, then $|S| \le (6\sqrt{d}/\varepsilon)^d$, where $\mathrm{cost}(\cdot) \in \{\mathrm{cost}_R, \mathrm{cost}_{DR}, \mathrm{cost}_D\}$.* □

Let $V = k \cdot (6\sqrt{d}/\varepsilon)^d$ the maximum number of cells that can contain points that belong to one of the k clusters. We observe that we can stop our procedure if the number of representatives is V. Thus, we can guarantee that the algorithm requires at most $V \cdot 3^d$ range queries.

Theorem 5 *There is an ε-tester for the radius clustering and diameter clustering problem that uses at most $k \cdot (18\sqrt{d}/\varepsilon)^d$ orthogonal range queries. There is an ε-tester for the discrete radius clustering problem that uses $k \cdot (162\sqrt{d}/\varepsilon)^d$ orthogonal range queries.* □

5.1 Dynamic Clustering

In this section we consider the problem of maintaining an approximate clustering of points in $\mathbb{R}^d$ under the operations *insert* and *delete*. Obviously, we can call the decision procedure from the previous section $\mathcal{O}(\log_{1+\varepsilon} B)$ times to find a clustering of size at most $(1+\varepsilon)B$ where B is the size of an optimal clustering and $B \geq 1$. When we combine this with a dynamic data structure that supports orthogonal range queries in time $A(n)$ (to report a single point in the query range) and update time $U(n)$ we immediately obtain the following result.

Corollary 5.2 *We can maintain an $(1+5\varepsilon)$ approximate radius/diameter clustering of a point set P in $\mathbb{R}^d$ (d constant) under the operations* insert *and* delete *in time $\mathcal{O}(U(n) + \log_{1+\varepsilon} B \cdot (A(n) \cdot k/\varepsilon^d + \exp(\mathcal{O}(k/\varepsilon^d)))$. If the parameters ε, d, and k are constants this is $\mathcal{O}(U(n) + A(n) + \log_{1+\varepsilon} B)$ time.*
□

Now, we want to obtain a time bound that is independent of the size of the clustering. We shall require an additional kind of oracle access: we allow the tester to query the oracle for the number of points within a certain range (this procedure could be also performed in our prior model in a logarithmic cost). We also need a procedure to compute a minimum (axis parallel) bounding box for the points inside a given cell. This can be easily done with $\mathcal{O}(d \log n)$ expected oracle accesses.

To avoid a simple binary search we use this bounding box. The size of the bounding box will always be the length of its longest side. Then we compute a clustering C for the current grid size (using the representatives for each cell). If l is the size of the largest bounding box of all grid cells, then we know that P can be clustered at cost at most $\mathrm{cost}(C) + 2 \cdot \sqrt{d} \cdot l$. Note that we can stop our process if $\frac{s}{\mathrm{cost}(C)} \leq \varepsilon/(3\sqrt{d})$ where s is the current size of the grid.

If we cannot stop we continue with a grid of size $l/2$. This way the number of grid cells with representatives is at most 3^d times the previous number of grid cells and there is at least 1 more such cell. We continue this procedure until we get a lower bound on the size of the current clustering. Then we have to do a logarithmic number of further steps and we are done.

Theorem 6 *We can maintain an $(1+\varepsilon)$ approximate radius/diameter clustering of a point set P in $\mathbb{R}^d$ (d constant) under the operations* insert *and* delete *in time $U(n) + \mathcal{O}(\exp(\mathcal{O}(V)) \cdot (k + \log(1/\varepsilon) + A(n) \cdot V \cdot \log n \cdot (k + \log(1/\varepsilon)))$. If k, d, and ε are constants then this is $\mathcal{O}(U(n) + A(n) \cdot \log n)$.* □

References

[1] P. K. Agarwal. Range searching. Chapter 31 in J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, CRC Press, 1997.

[2] P. K. Agarwal, C. M. Procopiuc. Exact and approximation algorithms for clustering. *Proc. 9th ACM-SIAM SODA*, pp. 658–667, 1998.

[3] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. *Proc. 41st IEEE FOCS*, pp. 240–250, 2000.

[4] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. *Proc. 40th IEEE FOCS*, pp. 656–666, 1999.

[5] P. Bose and P. Morin. Testing the quality of manufactured disks and cylinders. *Proc. 9th ISAAC*, pp. 129–138, 1998.

[6] P. Bose and P. Morin. Testing the quality of manufactures balls. *Proc. 6th WADS*, pp. 145–156, 1999.

[7] A. Czumaj, C. Sohler, and M. Ziegler. Property testing in computational geometry. *Proc. 8th ESA*, pp. 155–166, 2000.

[8] C. A. Duncan, M. T. Goodrich, and E. A. Ramos. Efficient approximation and optimization algorithms for computational metrology. *Proc. 8th ACM-SIAM SODA*, pp. 121–130, 1997.

[9] S. Doddi, M. Marathe, A. Mirzaian, B. Moret, and B. Zhu. Map labeling and its generalizations. *Proc. 8th ACM-SIAM SODA*, pp. 148-157, 1997.

[10] F. Ergün, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *JCSS*, 60:717–751, 2000.

[11] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. *Proc. 7th ACM SoCG*, pp. 281-288, 1991.

[12] A. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19:175–220, 1999.

[13] O. Goldreich. Combinatorial property testing (A survey). *Proc. DIMACS Workshop on Randomization Methods in Algorithm Design*, pp. 45–59, 1997.

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

[15] O. Goldreich and D. Ron. A sublinear bipartiteness tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.

[16] H. B. Hunt III, M. V. Marather, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.

[17] K. Mehlhorn, T. C. Shermer, and C. K. Yap. A complete roundness classification procedure. *Proc. 13th ACM SoCG*, pp. 129–138, 1997.

[18] D. Ron. Property testing. To appear in *Handobook of Randomized Algorithms*, P. M. Pardalos, S. Rajasekaran, J. Reif, and J. D. P. Rolim, eds., Kluwer Academic Publishers.

[19] C. Yap. Exact computational geometry and tolerancing metrology. In D. Avis and J. Bose, editors, *Snapshots of Computational and Discrete Geometry*, volume 3. McGill School of Computer Science, 1995. Technical Report SOCS-94.50.

[20] C. Yap and E.-C. Chang. Issues in the metrology of geometric tolerancing. *Proc. 2nd Workshop on Algorithmic Foundations of Robotics*, pp. 393–400, 1996.