Testing Properties of Sparse Images

Gilad Tsur
School of Electrical Engineering
Tel Aviv University
Email: giladt@post.tau.ac.il

Dana Ron
School of Electrical Engineering
Tel Aviv University
Email: danar@eng.tau.ac.il

***Abstract*—We initiate the study of testing properties of images that correspond to *sparse* $0/1$-valued matrices of size $n \times n$. Our study is related to but different from the study initiated by Raskhodnikova (*Proceedings of RANDOM, 2003*), where the images correspond to *dense* $0/1$-valued matrices. Specifically, while distance between images in the model studied by Raskhodnikova is the fraction of entries on which the images differ taken with respect to all n^2 entries, the distance measure in our model is defined by the fraction of such entries taken with respect to the actual number of 1's in the matrix. We study several natural properties: connectivity, convexity, monotonicity, and being a line. In all cases we give testing algorithms with sublinear complexity, and in some of the cases we also provide corresponding lower bounds.**

***Keywords*-Property Testing; Sublinear Algorithms; Images.**

I. Introduction

Suppose we are given access to an image that is defined by a $0/1$-valued $n \times n$ matrix M, and would like to know whether it has a particular property (e.g., the image it contains corresponds to a convex shape). We may read all pixels (bits) in the matrix and run an appropriate algorithm on this data, thus obtaining an exact answer in at least linear time. However, assume we are interested in a much more efficient algorithm, and are willing to sacrifice some precision. Namely, we seek a randomized, sublinear-time algorithm that can distinguish with high success probability between a matrix that has the specified property, and a matrix that is relatively far from having the property. In other words, we seek a *property testing* algorithm [24], [13].

In order to formalize the above question, we first need to define what it means to be *far* from having the property, and what access we have to the matrix. One natural definition of distance between matrices is the Hamming weight of their symmetric difference, normalized by the size of the matrices, which is n^2, and the most straightforward form of accessing the matrix is probing its entries. Indeed, this model of testing properties of images was introduced and studied by Raskhodnikova [22], and we later discuss in more details the results that she obtained as well as their relation to our results.

This work was supported by the Israel Science Foundation (grant number 246/08).

The abovementioned model is most suitable for relatively dense images, that is, images in which the number of 1-pixels (i.e., entries (i, j) for which $M[i, j] = 1$) is $\Omega(n^2)$. However, if the image is relatively sparse, e.g., the number of 1-pixels is $O(n)$, then a natural alternative is to normalize the distance with respect to the Hamming weight of the matrix, which we denote by $w(M)$, rather than to normalize by n^2. We believe that this type of measurement is appropriate in many contexts. For instance, an image of a single line of constant width is not generally viewed as very similar to an empty image, while it is considered so in the dense-images model. Essentially, while the dense-images model is suitable for testing images composed of areas, the sparse-images model works just as well with images composed of lines (or outlines).

An additional difference between the dense-images model and the sparse-images model is that in the latter model we also give the algorithm access to uniformly selected 1-pixels (in addition to query access to entries of its choice). Observe that if an image is sparse, then it actually makes sense to store it in a data-structure of size $O(w(M) \log n)$ rather than in an $n \times n$ matrix. Such space-efficient data structures may be easily devised to support uniform sampling of 1-pixels as well as answering queries to particular entries of M (possibly with an overhead of $O(\log w(M))$). In the dense image model (as in many property testing scenarios) the algorithm complexity is measured in terms of the number of queries it performs, where a query is checking whether the value of a pixel is 0 or 1. As we also allow our algorithms access to uniformly selected 1-pixels, we will wish to know how many of these were sampled, as well. Thus we consider two measures of complexity – the number of locations in the image that the algorithm queries, which we call *query complexity*, and the number of uniformly selected 1-pixels that the algorithm requests, which we call *sample complexity*.

We note that the relation between the dense-images model and the sparse-images model, is reminiscent of the relation between the dense-graphs model [13] and the sparse-graphs model [20], [17] (which extends the bounded-degree model [14]). We return to this relation subsequently, but first we state our results and the techniques we apply.

A. Our results

In what follows, when we use the term "complexity" we mean the sample and query complexity of the algorithm. For all the properties we study, except for line imprints, the running time of the algorithm is at most a logarithmic factor larger, and for line imprints it is polynomial in the number of queries (which is independent of n). The parameter ϵ is the *distance* parameter. Namely, the algorithm should accept with high constant probability[1] every matrix that has the property and should reject with high constant probability every matrix M that is ϵ-far from having the property (that is, more than $\epsilon \cdot w(M)$ pixels in M should be modified so that it obtains the property). We study the following properties.

- **Connectivity.** We say that an image M is *connected* if the underlying graph induced by the neighborhood relation between 1-pixels is connected. We give a testing algorithm for connectivity whose complexity is $\tilde{O}\left(\min\left\{w(M)^{1/2}, n^2/w(M)\right\} \cdot \epsilon^{-2}\right)$. Thus, as long as $w(M) \leq n^{4/3}$, the complexity of the algorithm increases like the square-root of $w(M)$, and once $w(M) > n^{4/3}$ it starts decreasing as $w(M)$ increases. We also prove a lower bound of $\Omega\left(\min\left\{w(M)^{1/3}, n^2/w(M)\right\}\right)$ for $\epsilon = \Theta(1)$. For one-sided error algorithms (that is, algorithms which accept an image with probability 1 if the image has the property) we show a simple lower bound of $\Omega(\min\{w(M), n^2/w(M)\})$ (which is $\Omega(w(M))$ for $w(M) = O(n)$).
- **A line (imprint).** We say that an image is a *line imprint* (or simply a *line*) if there exists a line such that all the pixels that the line intersects are 1-pixels, and there are no other 1-pixels in the image. We give a (one-sided error) algorithm for testing this property whose complexity is $O(\log(1/\epsilon)/\epsilon)$. This algorithm and its analysis are presented with a more general result concerning testing sparse images that have a small VC-dimension (and a corresponding result about learning when the distance measure is defined with respect to $w(M)$ rather than the size of the domain, which is n^2). While this result is fairly simple, it does not follow from the known transformation of (proper) learning results to testing results [13].
- **Convexity.** We say that an image M is *convex* if there exists a convex shape that is connected, closed and such that all the pixels that the shape intersects in M are 1-pixels, and there are no other 1-pixels in the image.[2] We assume without loss of generality that the convex shape is a polygon, and we consider a certain variant of this property where we require the gradient of the lines defining the convex shape to be of the form $1/r$ for an integer r.[3] For this property we give an algorithm whose complexity is $\tilde{O}(w(M)^{1/4} \cdot \epsilon^{-2})$.
- **Monotonicity.** We say that an image M is *monotone* if for every two 1-pixels (i_1, j_1) and (i_2, j_2), if $i_1 < i_2$, then $j_1 \leq j_2$. We give a one-sided error algorithm for testing monotonicity whose complexity is $\tilde{O}\left((n^{2/3}/w(M)^{1/3})\epsilon^{-2}\right)$. This algorithm improves on a simple sampling algorithm whose complexity is $O((w(M)/\epsilon)^{1/2})$, whenever $w(M) = \Omega(n^{4/5})$. (This simple algorithm only takes uniform samples and rejects if and only if it detects a violation of monotonicity.) For example, when $w(M) = \Theta(n)$, the dependence on n is reduced from $n^{1/2}$ to $n^{1/3}$. We also give an almost matching lower bound. Namely, we show that any (two-sided error) testing algorithm for monotonicity must have complexity $\Omega(\min\{w(M)^{1/2}, n^{2/3}/w(M)^{1/3}\})$ (for constant ϵ).

For illustrations of the different properties, see Figure 1. Our algorithms (with the exception of the line imprint algorithm) are assumed to be given a constant factor estimate of $w(M)$. The lower bounds hold when the algorithm has such knowledge as well. In the case of connectivity and convexity we show how such an estimate can be obtained without increasing the complexity of the algorithm.

B. Techniques

As noted in the previous subsection, one of our results, concerning testing the basic property of being a line, is part of a more general technique that exploits the small VC-dimension of the property. While using bounds on the VC-dimension is far from being new, there is a small "twist" in our application. The other results differ from this one, and though each has its own particularities, they can be viewed as sharing a common "theme".

This common theme is that the image is considered in two "scales": a coarser one and a finer one. The coarser scale is determined by uniform samples of 1-pixels, and the finer scale by queries. For example, in the case of testing connectivity, the algorithm considers a partition of the input matrix into submatrices (of size roughly $\sqrt{w(M)}$ in each dimension). Given a sample of 1-pixels, the algorithm first checks whether the submatrices that contain samples are connected. If they are not connected, then the algorithm rejects.[4] Otherwise these submatrices are considered the *backbone* of the image and the algorithm now tests (with the use of queries), whether all but a small fraction of the submatrices in the backbone are "internally connected",

[1] Whenever we refer to an event that occurs "with high constant probability", we mean with probability at least $1 - \delta$ for any small constant δ of our choice.

[2] We assume that the shape is contained within the image area.

[3] The requirement for a gradient of the form $1/r$ is imposed to avoid a host of issues relating to the irregular shape of pixelated lines in polygons with other gradients. It is possible that our results can be extended to such cases almost unchanged or that the difference is of essence.

[4] Indeed, this causes the algorithm to have two-sided error. As noted previously, if we require that the algorithm have one-sided error, then there is no sublinear algorithm when the matrix is relatively sparse.)

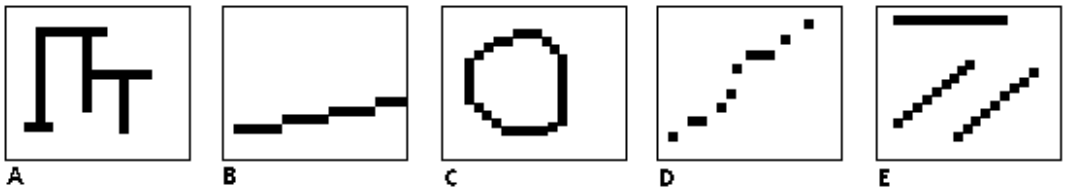

Figure 1. Examples of the tested properties in images: (A) is a connected image, (B) is a line imprint, (C) is convex and (D) is monotone. The image (E) is relatively far from having any of the properties.

and that all but a small fraction of the points are "well connected" to them.

In the case of monotonicity there is no predetermined partition, but rather the sample determines such a partition (or causes rejection since a violation of monotonicity is observed). The partition is such that if the image is far from being monotone, then (almost all) the violations are within the submatrices defined by the sample. We then show that by performing queries within these submatrices (with an appropriate distribution over the queries), we will detect a violation with high probability. The convexity testing algorithm also does not have a predetermined partition, but its use of queries is more similar to the connectivity testing algorithm (though it is able to exploit certain useful features of convex images, and is hence more efficient).

C. The work of Raskhodnikova [22] and its relation to our work

Raskhodnikova studies three properties in the dense-images model: *connectivity*, *convexity*, and *being a half-plane*. All the algorithms in [22] have complexity that is at most quadratic in $1/\epsilon$, and have no dependence on the size (n^2) of the matrix.

We also consider the (same) property of connectivity and give two algorithms. The first is more efficient when the matrix is below a certain threshold of the density (i.e., $w(M) \leq n^{4/3}$) and the second is more efficient when the density goes above this threshold. The second algorithm is essentially the same as the connectivity testing algorithm in [22] (with the appropriate setting of certain parameters), and its analysis is similar (with certain subtleties). This algorithm is also similar to the connectivity testing algorithm in bounded-degree graphs [14] (though the analysis is naturally different).

We also have an algorithm for testing convexity, however, the notion we study of convexity (appropriate for sparse images where $w(M) = O(n)$) and the one studied in [22] (which considers the convex hull of 1-pixels and hence is appropriate for dense images) are different, and so the results are incomparable.

Finally, our testing algorithm for a line imprint can be seen as the "sparse analog" of being a half-plane. As noted by Raskhodnikova [22], it is possible to test whether an image corresponds to a half-plane by attempting to learn the half-plane. She suggests a direct approach that is more efficient (the complexity is $O(1/\epsilon)$ rather than $O(\log(1/\epsilon)/\epsilon)$). For the line imprint we do take what can be seen as a "learning-based" approach (for an appropriate notion of learning), and it is possible that in our case an improvement is possible as well by a direct approach.

D. The relation to models for testing graph properties

In the adjacency-matrix (dense-graphs) model [13], the distance between two n-vertex graphs is the fraction of entries on which their adjacency matrices differ (where the fraction is with respect to n^2). In this model the algorithm is allowed to probe the matrix (that is, query whether there is an edge between any pair of vertices of its choice). In the sparse/general graphs model [20], [17], distance is measured with respect to the number of edges, m, in the graph (or a given upper bound on this number). The algorithm may query any vertex of its choice on its i^{th} neighbor (for any i), and it may also query whether there is an edge between any two vertices (the latter is useful when the graph is sufficiently dense).

Thus there is a similarity in the way the sparse/general graphs model relates to the dense-graphs model and the way the sparse-images model relates to the dense-images model. We also note that for both types of objects (graphs and images) while some properties are meaningful only in one model (dense or sparse), there are properties that are of interest in both models. For example, in the case of graphs, the property of *bipartiteness* (studied in [13], [2], [14], [17]) exhibits an interesting behavior when considering the whole spectrum of graph densities. In particular, as long as the number of edges, m in the graph is below $n^{3/2}$, the complexity grows like $n^{1/2}$, and once the edge-density increases, the complexity behaves like n^2/m (and there is an almost tight corresponding lower bound). In the case of images, the property of connectivity exhibits a somewhat similar behavior (as discussed previously).

E. Other related work

In addition to the work of Raskhodnikova [22] (which has been discussed above), Kleiner et al. [18] study testing partitioning properties of dense images, and there have been quite a few works on testing geometric properties, and in

particular convexity, in various models [9], [21], [5], [7], [6]. The property of monotonicity has been studied quite extensively in the context of functions [9], [3], [12], [8], [1], [15], [11], [10], [16], [4], [19].

F. Organization

Due to space constraints we have chosen to mainly focus on one property - connectivity. For the other properties we provide a short overview. All missing details can be found in the full version of this paper [23].

II. Testing Connectivity

For a $\{0,1\}$ matrix M, consider the underlying undirected graph $G(M) = (V(M), E(M))$, where $V(M) = \{(i,j) : M[i,j] = 1\}$ (so that $|V(M)| = w(M)$) and $E(M)$ consists of all pairs $(i_1, j_1) \neq (i_2, j_2)$ in $V(M)$ such that $|i_1 - i_2| \leq 1$ and $|j_1 - j_2| \leq 1$. We say that M is *connected* if the underlying graph $G(M)$ is connected. Given the correspondence between M and $G(M)$ we shall interchangeably refer to 1-pixels in M and vertices of $G(M)$. We shall assume that $w(M) \geq 1$, since we can detect that $w(M) = 0$ by asking for a single sample 1-pixel (and getting none), in which case we can accept. For the sake of simplicity, unless it affects the analysis (by more than introducing constant factors in the complexity), we ignore floors and ceilings. Note that as defined above, two pixels that are neighbors on a diagonal are considered neighbors.

In this section we describe an algorithm for testing connectivity whose sample complexity, query complexity and running time are $\tilde{O}\left(\min\left\{w(M)^{1/2}, n^2/w(M)\right\}\right) \cdot \mathrm{poly}(1/\epsilon)$. Thus, as long as $w(M) \leq n^{4/3}$, the complexity increases with $w(M)^{1/2}$, and once $w(M)$ goes above $n^{4/3}$, the complexity starts decreasing. We later prove a lower bound of $\Omega(\min\{w(M)^{1/3}, n^2/w(M)\})$ queries (for a constant ϵ) on the complexity of any two-sided error algorithm. We also show that if one requires that the algorithm have one-sided error, then there is no sublinear-time algorithm (that is, there is a lower bound of $\Omega(w(M))$).

A. The algorithm

We start by assuming that we are given a constant factor estimate, $\widehat{w}$ of $w(M)$. We shortly discuss in Subsection II-A2 how this assumption can be removed.

We describe an algorithm such that given $w(M)/c \leq \widehat{w} \leq c \cdot w(M)$ (for some fixed and known constant c), the algorithm has query and sample complexities, as well as running time, $\tilde{O}(\sqrt{w(M)}\epsilon^{-2})$. We may thus assume that $\epsilon = \omega(1/\sqrt{w(M)})$ or else, we can take a single sample 1-pixel, run a Breadth First Search (BFS) to find its connected component in $G(M)$ by performing $O(1/\epsilon^2) = O(w(M))$ queries, and then take an additional sample of $\Theta(1/\epsilon)$ 1-pixels to verify that are are no (few) 1-pixels that do not belong to this component. In Subsection II-A1 we shortly discuss the case that $w(M)$ is relatively large (the threshold is roughly around $w(M) = n^{4/3}$). In this case, an alternative (and simpler) algorithm, which generalizes the algorithm in [22], has better performance (and in particular improves as $w(M)$ increases).

The high level idea of the algorithm is as follows: the algorithm tries to find evidence that the tested matrix M is not connected, where the evidence comes in one of the following two forms. (1) "Hard" evidence, in the form of a small connected component in $G(M)$; (2) "Soft" ("statistical") evidence in the form of more than one connected component when viewing the matrix at a "coarser" resolution. Namely, if we partition the matrix into (equal-size) submatrices, and take a sample of 1-pixels, then we can define a graph over those submatrices that contain at least one sample 1-pixel similarly to the way it was defined for single 1-pixels (i.e., $G(M)$). The algorithm checks whether this "backbone' graph is connected. Evidence against connectivity of this type is "soft", or "statistical" since it is possible that the matrix is connected but the sample missed some submatrix, causing the backbone graph to be disconnected. Basing the decision of the algorithm on the second type of evidence and not only on the first, is what makes the algorithm have two-sided error. Evidence of the "hard" form is obtained by performing several BFS's on $G(M)$ (note that the neighbors of a vertex in $G(M)$ that corresponds to an entry (i,j) in M can be obtained by performing 8 queries to M).

Algorithm 1: **Testing connectivity (Version I)**

1) Consider a fixed partition of M into equal-size submatrices of dimensions $s \times s$ where $s = \sqrt{\widehat{w}/c}$ (recall that $\widehat{w} \leq c \cdot w(M)$ and that the constant c is known to the algorithm).
2) Take a sample S_1 of $t_1 = \Theta\left(\sqrt{\widehat{w}} \cdot \log(\widehat{w})\right)$ uniformly distributed 1-pixels in M and consider all non-empty submatrices in the abovementioned partition (that is, all submatrices that contain a sample 1-pixel). Let $B(S_1)$ be the ("backbone") graph whose vertices are the non-empty submatrices, and where there is an edge between two submatrices if they are adjacent (horizontally, vertically, or diagonally). If $B(S_1)$ is not connected, then REJECT (otherwise, continue).
3) Select, uniformly at random, $t_2 = \Theta(\log(\widehat{w})/\epsilon)$ non-empty submatrices (vertices in $B(S_1)$). For each submatrix selected, consider the first sample 1-pixel that fell into the submatrix, and perform a BFS in $G(M)$ starting from the vertex that correspond to this 1-pixel. Stop once the BFS reaches at least $8\sqrt{c \cdot \widehat{w}}/\epsilon$ vertices in $G(M)$ or the BFS gets "stuck" (a small connected component in $G(M)$ is detected). In the latter case REJECT (otherwise, continue).
4) Take an additional sample, S_3, of $t_3 = \Theta(1/\epsilon)$ 1-pixels. If any selected 1-pixel belongs to a submatrix that does not neighbor a submatrix in the backbone (a vertex of $B(S_1)$), then REJECT. Otherwise, perform a BFS starting from each sample 1-pixel in S_3 as

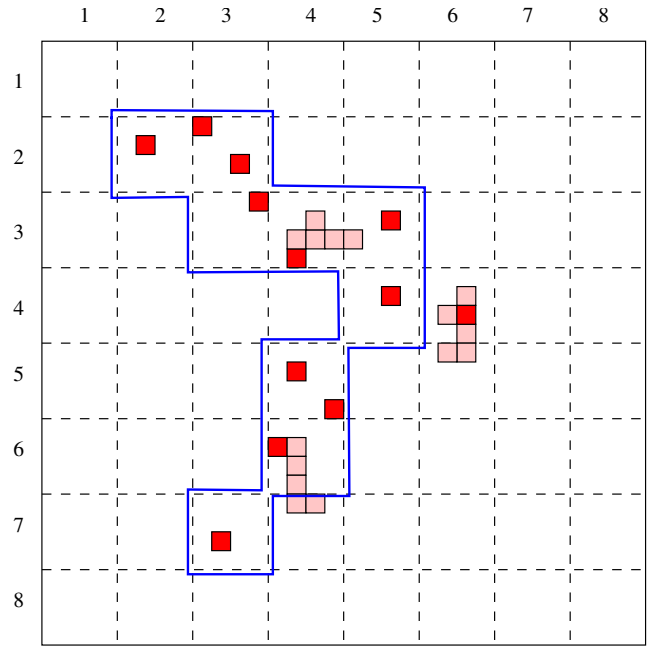

Figure 2. An illustration for the execution of Algorithm 1. The partition into submatrices is marked by a grid of dashed lines. The sampled 1-pixels (in either Step 3 or in Step 4) are marked by dark filled pixels, and the queried entries that are answered by 1 in the course of the BFS's are marked by lighter filled pixels. The backbone is outlined by a bold line. Note that the marked pixels outside the backbone correspond to a BFS performed in Step 4.

described in the previous step. If a small connected component is found then REJECT.

5) If no step caused rejection, then ACCEPT.

For an illustration of a (successful) execution of the algorithm, see Figure 2.

Theorem 2.1: Algorithm 1 is a (two-sided error) testing algorithm for connectivity. Its sample and query complexities as well as its running time are $\tilde{O}\left(\sqrt{\widehat{w}}\cdot\epsilon^{-2}\right) = \tilde{O}\left(\sqrt{w(M)}\cdot\epsilon^{-2}\right)$.

Proof: The sample complexity of the algorithm is $t_1+t_3 = O\left(\sqrt{\widehat{w}}\cdot\log(\widehat{w})\cdot\epsilon^{-1}\right)$. Its query complexity is $O(\log(\widehat{w})\cdot\epsilon^{-1})\cdot O(\sqrt{\widehat{w}}\cdot\epsilon^{-1}) = O\left(\sqrt{\widehat{w}}\cdot\log(\widehat{w})\cdot\epsilon^{-2}\right)$, since it performs $t_2+t_3 = O(\log(\widehat{w})\cdot\epsilon^{-1})$ searches (in Steps 3 and 4), in each search it reaches $O(\sqrt{\widehat{w}}\cdot\epsilon^{-1})$ vertices in $G(M)$, and determining all neighbors of a vertex in $G(M)$ can be done by performing 8 queries to M. The algorithm can be implemented so that it run in time $\tilde{O}\left(\sqrt{\widehat{w}}\cdot\epsilon^{-2}\right)$.

We turn to analyzing the correctness of the algorithm. In all that follows, when we refer to submatrices, we mean one of the $(n/s)^2 = c\cdot n^2/\widehat{w}$ submatrices in the partition defined by the algorithm. For each $s\times s$ submatrix, we say that the submatrix is *heavy* if the number of 1-pixels in it is at least $s/2$ (recall that $s=\sqrt{\widehat{w}/c}$). Otherwise it is *light*. By our choice of the sample size $t_1 = \Theta(\sqrt{\widehat{w}}\log(\widehat{w}))$, with high constant probability, in Step 2 we'll get at least one sample 1-pixel from each heavy submatrix. We shall say in such a case that the sample S_1 is *typical*.

Consider first the case that M is connected. We claim that the probability that it is rejected is at most a small constant. First we observe that M cannot be rejected due to a small connected component of $G(M)$ being found in Step 3 or Step 4, since $G(M)$ consists of a single connected component of size $w(M)$ (and we assume that $\epsilon=\omega(1/\sqrt{\widehat{w}})=\omega(1/\sqrt{w(M)})$. Next we note that if S_1 is typical, then the backbone graph $B(S_1)$ must be connected. This is true since each submatrix is of size $s\times s$ for $s=\sqrt{\widehat{w}/c}$, so the existence of more than one connected component in $B(S_1)$ implies that some heavy submatrix was not hit (under the premise that M is connected).[5] By the same reasoning, if S_1 is typical, then all 1-pixels in the sample selected in Step 4 belong to submatrices that belong to or neighbor the backbone submatrices. Since the probability that S_1 is not typical is upper bounded by a small constant, M will be accepted with high constant probability.

We now turn to deal with the case that M is ϵ-far from being connected. If the backbone graph $B(S_1)$ is not connected, then M is rejected in Step 2 of the algorithm, so we consider the case that $B(S_1)$ is connected. We say that a submatrix in the backbone is "reliable" if it would

[5]To see this, consider the 1-pixels in the area of two connected components in $B(S_1)$, C_1 and C_2. As $G(M)$ is connected, there must be at least one path between these pixels in $G(M)$, and as $B(S_1)$ isn't connected, the length of this path must be at least s. Let us trace this path until it leaves the submatrices neighboring C_1. In particular, consider the last s 1-pixels in this subpath. These pixels pass through no more than 2 submatrices and so one of these is heavy. Such a heavy submatrix must be hit by S_1 in a typical sample.

pass the BFS test performed by the algorithm in Step 3 (starting from the first sample 1-pixel that fell into it). Otherwise it is "unreliable". Similarly, we say that a 1-pixel is "well connected" to the backbone if it would pass the test performed in Step 4 of the algorithm (that is, it belongs to a submatrix that neighbors one of the backbone submatrices, and a BFS that starts from it does not detect a small connected component).

Suppose that the number of submatrices in the backbone that are unreliable is greater than $(\epsilon/8)\sqrt{w(M)} \geq (\epsilon/8) \cdot \sqrt{\widehat{w}/c}$. Recall that the total number of submatrices in the backbone is at most $t_1 = \Theta\left(\sqrt{\widehat{w}} \cdot \log(\widehat{w})\right)$. Therefore, one of these submatrices is selected with high constant probability in Step 3 of the algorithm (where $\Theta\left(\log(\widehat{w}/\epsilon)\right)$ submatrices of the backbone are selected), causing the algorithm to reject. Similarly, if the fraction of 1-pixels that are not well-connected to the backbone is greater than $\epsilon/8$, then with high constant probability we'll obtain such a 1-pixel in Step 4 of the algorithm and reject.

We next show that if both the number of unreliable submatrices in the backbone is at most $(\epsilon/8)\sqrt{w(M)}$, and the fraction of 1-pixels that are not well-connected to the backbone is at most $\epsilon/8$, then M is ϵ-close to being connected. We show this by describing how M can be made connected with relatively few modifications, building on the backbone. This implies that if M is ϵ-far from being connected then it will be rejected with high constant probability. Details follow.

For each of the reliable submatrices of the backbone, consider the BFS performed starting from the first sample 1-pixel in S_1 that belongs to the submatrix. Since the submatrix is reliable, at least $8\sqrt{c\widehat{w}}/\epsilon$ vertices in $G(M)$ are reached by the BFS. Similarly, for each well-connected 1-pixel, consider the BFS that starts from this 1-pixel and reaches at least $8\sqrt{c\widehat{w}}/\epsilon$ vertices in $G(M)$. Since the total number of vertices in $G(M)$ is $w(M)$, the number of connected components in the subgraph of $G(M)$ that is induced by the union of all these BFS's is at most $\frac{w(M)}{8\sqrt{c\widehat{w}}/\epsilon} \leq (\epsilon/8)\sqrt{w(M)}$. We note that, by their definition, well-connected 1-pixels may belong to submatrices in the backbone. That is, there may be more than one BFS that starts in the same submatrix.

Next we deal with the unreliable submatrices in the backbone (where there are at most $(\epsilon/8)\sqrt{w(M)} \leq (\epsilon/8)w(M)/s$ such submatrices). For each unreliable submatrix in the backbone, we change at most s of the entries in it from 0 to 1, so as to obtain a connected component that corresponds to some arbitrary row in the submatrix (say, the middle row). Let M' be the resulting matrix (where M' and M differ in at most $(\epsilon/8)w(M)$ entries).

At this point we have at most $(\epsilon/4)\sqrt{w(M)}$ connected components in the subgraph of $G(M')$ that is induced by the aforementioned BFS's and the modified entries in the unreliable submatrices. These components intersect all submatrices in the backbone and possibly additional neighboring submatrices (due the the BFS's that start from well-connected 1-pixels). If we consider an auxiliary graph whose vertices are these components and where there is an edge between two components if they intersect neighboring submatrices or the same submatrix, then this auxiliary graph is connected. Let T be some (arbitrary) spanning tree of this auxiliary graph. For each edge in the spanning tree (a pair of neighboring (or identical) submatrices that are intersected by different connected components), we can modify at most $2^{3/2}s = 2^{3/2}\sqrt{\widehat{w}/c} \leq 2^{3/2}\sqrt{w(M)}$ entries in the neighboring (or identical) submatrices from 0 to 1 so as to connect the two corresponding connected components, and get a single connected component. Let M'' be the resulting matrix (so that M'' and M' differ on less than $(3\epsilon/4)w(M)$ entries).

Finally, we observe that all vertices in $G(M'')$ that do not belong to the abovementioned single connected component in $G(M'')$ necessarily correspond to 1-pixels in M that are not well-connected (though it is possible that some 1-pixels that are not defined as well-connected belong to the single connected component). Therefore, we can change all these (at most $(\epsilon/8) \cdot w(M)$) entries in M'' from 1 to 0 and remain with a single connected component in the resulting matrix. The total number of modifications made is at most $(\epsilon/8)w(M) + (3\epsilon/4)w(M) + (\epsilon/8) \cdot w(M) = \epsilon w(M)$, implying that M is ϵ-close to being connected, as claimed. ■

1) An alternative algorithm for dense submatrices: The alternative algorithm has complexity $O(n^2/w(M)) \cdot \mathrm{poly}(1/\epsilon)$, so that it improves on Algorithm 1 when $w(M) = \Omega(n^{4/3})$. The algorithm is essentially a generalization of the algorithm of Raskhodnikova [22] for testing connectivity in the dense-images model (which is appropriate when $w(M) = \Theta(n^2)$) and is reminiscent of the algorithm for testing connectivity in bounded-degree graphs [14]. The algorithm simply takes a sample of $\Theta(1/\epsilon)$ 1-pixels and from each it performs a BFS in $G(M)$ until it reaches a sufficiently large number of vertices (of the order of $(n^2/w(M))\epsilon^{-2}$), or it finds a small connected component, in which case it rejects.

2) Obtaining an estimate of $w(M)$: It is possible to obtain a constant factor estimate of $w(M)$ by taking a sample of size $O(\min\{\sqrt{w(M)}, n^2/w(M)\})$ and performing at most these many queries. The idea behind the algorithm is the following. It is possible to obtain an estimate of $w(M)$ in two different ways. Roughly speaking, the first achieves a better performance when $w(M)$ is "large", and the second when $w(M)$ is "small". Specifically, the first approach is to simply query uniformly selected entries in M and take the fraction of entries that are 1 (among those queried) to be an estimate for $p(M) \stackrel{\text{def}}{=} w(M)/n^2$, and the second is to estimate the collision probability when taking

uniform samples. By combining the two procedures we get the desired estimate (with the stated complexity).

B. Lower bounds on testing connectivity

We start with a simple lower bounded for one-sided error algorithms, and then turn to two-sided error algorithms.

Theorem 2.2: Any one-sided error testing algorithm for connectivity must perform $\Omega(\min\{w(M), n^2/w(M)\})$ queries (for a constant ϵ). The lower bound holds when the algorithm is given an estimate $\widehat{w}$ such that $w(M)/2 \leq \widehat{w} \leq 2w(M)$.

The proof of Theorem 2.2 can be found in the full version of this paper [23].

Theorem 2.3: Any (two-sided) error testing algorithm for connectivity has sample complexity and query complexity $\Omega(\min\{w(M)^{1/3}, n^2/w(M)\})$ (for a constant ϵ). The lower bound holds when the algorithm is given an estimate $\widehat{w}$ such that $w(M)/2 \leq \widehat{w} \leq 2w(M)$.

Proof: Here we establish the claim for $\widehat{w} = \Theta(n)$ (where the bound is $\Omega(n^{1/3})$). in the fulll version of this paper [23] we explain how to modify it to smaller and larger values of $\widehat{w}$ (and hence $w(M)$). In order to prove the lower bound we define two families of matrices. In the first family, denoted $\mathcal{F}_1$, all matrices are connected, and in the second family, denoted $\mathcal{F}_2$, with very high probability over the choice of a random matrix in $\mathcal{F}_2$, the matrix is $\Omega(1)$-far from being connected. We shall show that any algorithm that takes a sample of size $o(n^{1/3})$ and performs at most these many queries, cannot distinguish with constant probability between a matrix selected uniformly at random from $\mathcal{F}_1$ and a matrix selected uniformly at random from $\mathcal{F}_2$.

Defining the two families. Consider a partition of the entries of an $n \times n$ matrix into submatrices of dimensions $n^{1/3} \times n^{1/3}$. For both families there will actually be 1-pixels only in the first "row" of these submatrices, where we number this sequence of submatrices from 1 to $n^{2/3}$ (from left to right).

Each matrix in $\mathcal{F}_1$ is determined by $2n^{2/3}$ integers, $i_1, \ldots, i_{n^{2/3}}$ and $j_1, \ldots, j_{n^{2/3}}$, where $1 \leq i_k, j_k \leq n^{1/3}$ for every $1 \leq k \leq n^{1/3}$. These integers determine the locations of the 1-pixels in the matrix in the following way: For each k, there are 1-pixels in row i_k of submatrix number k and submatrix number $k+1$ (if such exists), and there are 1-pixels in column j_k of submatrix number k. All other entries are 0.

Each matrix in $\mathcal{F}_2$ is determined by two subsets $T_r, T_c \subset [n^{2/3}]$ each of size $n^{2/3}/2$ and by $n^{2/3}$ indices $\{i_k\}_{k \in T_r} \cup \{j_\ell\}_{\ell \in T_c}$ where $1 \leq i_k, j_\ell \leq n^{1/3}$ for every $k \in T_r$ and $\ell \in T_c$. These integers determine the locations of the 1-pixels in the matrix in the same way that was defined for matrices in $\mathcal{F}_1$. That is, For each $k \in T_r$, there are 1-pixels in row i_k of submatrix number k and submatrix number $k+1$ (if such exists), and for each $\ell \in T_c$ there are 1-pixels in column j_ℓ of submatrix number ℓ. All other entries are 0. Thus, the difference between matrices in $\mathcal{F}_1$ and matrices in $\mathcal{F}_2$, is that in the latter family, there may be submatrices (in the first row of submatrices) that are "empty" (contain only 0's) or contain only a row of 1-pixels and no column of 1-pixels.

Properties of the two families. By the above description, every matrix in $\mathcal{F}_1$ is connected. On the other hand, it is not hard to verify that with very high probability (at least $1 - \exp(-n^\alpha)$ for some constant $\alpha > 0$), a uniformly selected matrix in $\mathcal{F}_2$ will be $\Omega(1)$-far from being connected. The reason is that with high probability over the choice of M in $\mathcal{F}_2$, the graph $G(M)$ will contain $\Omega(n^{2/3})$ connected components, and furthermore, if we consider these components from left to right, then a constant fraction of pairs of consecutive connected components are separated by a submatrix that contains no 1-pixels.

The difficulty of distinguishing between the two families. Consider any (two-sided error) algorithm for testing connectivity of matrices that takes a sample of $o(n^{1/3})$ 1-pixels and asks $o(n^{1/3})$ queries. We may assume without loss of generality that it first takes the sample and then performs all queries (possibly adaptively). We first show that for both families, the distributions on sampled 1-pixels are very similar. More precisely, we show that unless a certain low probability event occurs, the distributions on samples are identical. We later deal with the answers to queries.

Since once the algorithm is given a sample 1-pixel it can determine in an additional constant number of queries whether the 1-pixel belongs to a row or to a column (and in the former case whether the row extends to the next submatrix or to the previous submatrix), we assume that the algorithm is actually given a sample of rows/columns. That is, each sample is either of the form (k, i_k) or (ℓ, j_ℓ) for $k, \ell \in [n^{2/3}]$ and $i_k, j_k \in [n^{1/3}]$. Rather than first selecting a matrix uniformly from $\mathcal{F}_1$ (similarly, uniformly from $\mathcal{F}_2$) and then generating a sample, we may think of the sample being generated in the process of determining the uniformly selected matrix. Specifically, for each family we define a "process" ($\mathcal{P}_1$ for $\mathcal{F}_1$ and $\mathcal{P}_2$ for $\mathcal{F}_2$) that generates samples that are distributed according to a uniformly selected matrix in the family, while constructing the matrix. This is done as follows.

For each new sample, both of the processes first flip a coin with bias $2/3$ to decide whether to generate a row or a column (as in both families the number of 1-pixels that belong to rows is twice as large as the number of 1-pixels that belong to columns). Suppose that a row is to be generated (the generation of a column is analogous). Let t be the number of different rows already generated (where by our assumption on the sample complexity of the algorithm, $t = o(n^{1/3})$). Then the process $\mathcal{P}_1$ flips a coin with bias $t/n^{2/3}$ to determine whether the new row will be identical to a row that already appeared in the sample. If the coin turns

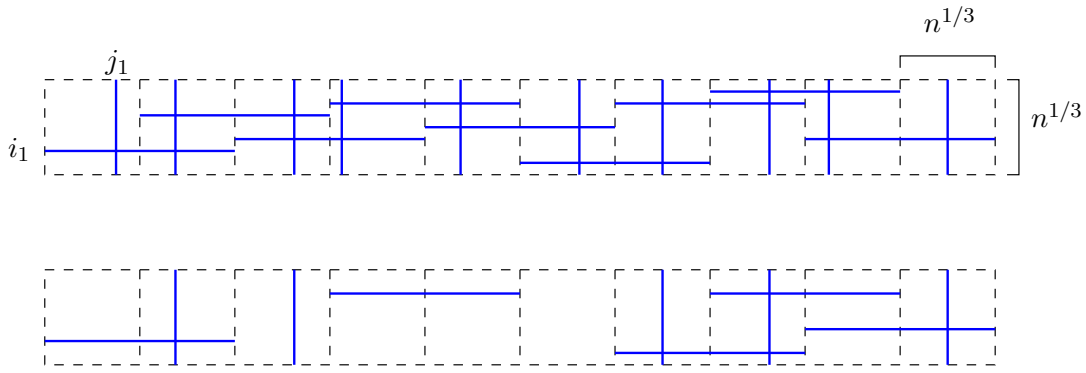

Figure 3. An illustration for the proof of Theorem 2.3 for $\widehat{w} = n$. On the top is an example of the first row of submatrices of a matrix in $\mathcal{F}_1$, and on the bottom is an example of the first row of submatrices of a matrix in $\mathcal{F}_2$. The outline of the submatrices is marked in dashed lines, and the rows and columns of 1-pixels are marked by bold lines.

out "heads", then one of the previously generated rows is selected to be the next sample row, while if the coin turns out "tails", then the process uniformly selects a submatrix k that is not yet associated with a row (that is, there is no row starting at this submatrix and ending in the next). It then uniformly selects $i_k \in [n^{1/3}]$ to determine the position of the row in submatrix k (and $k+1$). The process $\mathcal{P}_2$ does the same *except* that it flips a coin with bias $t/(n^{2/3}/2) = 2t/n^{2/3}$. The important observation is that for both processes, for any choice of a prefix of the sample, conditioned on the coin coming up "tails" (which occurs with probability at least $1 - 2t/n^{2/3}$ in both cases), the distribution over the new row is *identical*.

The above discussion implies, that, since the algorithm takes a sample of size $o(n^{1/3})$, with probability at least $1 - o(n^{1/3}) \cdot o(n^{1/3})/n^{2/3} = 1 - o(1)$, the distributions over the samples that the algorithm observes are identical for both families (processes).

It remains to deal with the queries. Here the argument is even simpler, where we now let the two processes answer queries while continuing to construct a matrix in their respective families. We may assume, without loss of generality, that the algorithm does not ask queries about 1-pixels that belong to rows/columns that it observed in the sample (as we already gave the algorithm the complete row/column "for free"). Thus the algorithm only asks queries about entries that do not belong to sample rows/columns. We claim that given that the algorithm asks $o(n^{1/3})$ queries, for both processes (families), with probability $1 - o(1)$, all queries are answered by 0.

To verify this, consider any fixed submatrix k. If the algorithm asked already t queries in the submatrix, so that the queries belong to at most t rows and at most t columns, and they were all answered by 0, for both processes, the probability that the next query in the submatrix is answered by 1 is upper bounded by $O(1/(n^{1/3} - t))$. Since the algorithm performs $o(n^{1/3})$ queries, the probability that it gets an answer of 1 to any of its queries, is $o(1)$, as claimed.

Thus, if the algorithm takes a sample of size $o(n^{1/3})$ and performs $o(n^{1/3})$ queries, then with probability at least $1 - o(1)$ the distributions on samples and the answers to its queries are identical when the matrix is uniformly selected in $\mathcal{F}_1$ and when it is uniformly selected in $\mathcal{F}_2$. This implies that there is no testing algorithm with this complexity for the property of connectivity when $w(M) = \Theta(n)$. ■

III. A Short Overview of the Other Results

In the following subsections we give a flavor of the testing algorithms for the other properties we study. Unless stated otherwise, assume that w is the number of pixels in the $n \times n$ image M that we are testing. Likewise, assume that ϵ is fixed.

A. Testing and Learning in the Sparse Image Model

In our Testing and Learning results we show that membership in families of images with a small VC-dimension [25] is easy to test for. It was observed in [13], that given a proper learning algorithm for a class of Boolean functions $\mathcal{C}$ (which is allowed queries and works under the uniform distribution), we can easily transform it into a testing algorithm for the property of membership in this class, with the same complexity. As the sparse image model is not a classic learning setting, we adapt results that show that classes of functions with small VC-dimension are easy to learn (and thus, to test for) in our model. The approach is as follows.

Imagine we wish to test for membership in a class of images $\mathcal{C}$. We begin by assuming that the number of 1-pixels, w in the tested image, M, is known to us. We can thus think of learning this image only from the members of $\mathcal{C}$ that are of size exactly w. We note that if an image $I \in \mathcal{C}$ that is of size w is different from M by ϵw pixels, then it is different from M by at least $\epsilon w/2$ pixels that are 1-pixels in M. Thus, based on the standard results from learning theory, given the exact value of w we can test for

membership in a number of queries that depends only on the VC-dimension of $\mathcal{C}$. We then extend this result to the case where w is unknown. We present a procedure that performs a type of binary search for w while learning, and that uses queries and not only uniform samples of 1-pixels from M. This costs us an additional complexity factor of $\tilde{O}(\log(n))$.

Finally, we give an algorithm that tests for a particular property - being the imprint of a line, in a number of queries that depends only on ϵ. As stated in the introduction we say that an image is a *line imprint* (or simply a *line*) if there exists a line such that all the pixels that the line intersects are 1-pixels, and there are no other 1-pixels in the image. This algorithm uses the constant VC-dimension of lines but does not require the additional $\tilde{O}(\log(n))$ factor we generally pay for not knowing what w is. It seems reasonable that other such algorithms can be designed for additional properties.

We note that the results and principles we present here may be of general interest in learning and testing sparse sets. If we consider an image as the set of 1-pixels it contains, we only use the limited VC-dimension and the ability to query uniform members of the set.

B. Testing Convexity

As noted in the introduction, we say that an image M is *convex* if there exists a convex shape that is connected, closed and such that all the pixels that the shape intersects in M are 1-pixels, and there are no other 1-pixels in the image. We assume without loss of generality that the convex shape is a polygon, and we consider a slightly restricted version of this property where we require the slope of the lines defining the convex shape to be of the form $1/r$ for an integer r. This leads us to an alternative definition of a convex shape that is based on "blocks" and "sub-blocks". Informally, a sub-block is a set of pixels that appear consecutively in the same row or column. A block is a connected series of sub-blocks that all have the same number of pixels (which we call the "period length" of the block). It turns out that a convex shape is an image such that all the top-most, bottom-most, left-most and right-most 1-pixels are (within each set) connected, and such that these sets of pixels are connected between them in a structured manner. For instance, The top-most 1-pixel in the leftmost set of pixels and the left-most 1-pixel in the topmost set of pixels are connected by a series of vertical blocks of monotonically decreasing period-length followed by a series of horizontal blocks of monotonically increasing period-length.

We show that there are at most $O(\sqrt{w})$ different blocks in a convex image, and thus reach the conclusion that it can be learned almost exactly using $\tilde{O}(\sqrt{w})$ queries (the boundaries of each block can be learned using $O(\log(w))$ queries). This immediately leads to a $\tilde{O}(\sqrt{w})$ testing algorithm, but we show that we can improve on this and get a $\tilde{O}(w^{1/4}))$ algorithm. This algorithm uses uniform sampling of 1-pixels to get a sample of the different blocks in M, and uses both queries and samples to check that a sample of these blocks is "properly connected" (that is, that the interval between two consecutive blocks is indeed composed of blocks consistent with a convex shape). Some technical adjustments are made to this approach as we see that we can't always perform such a verification. Essentially, we show that the number of pixels in intervals we can't properly check is not expected to be very large.

C. Testing Monotonicity

We say that a matrix M is *monotone* if for every two 1-pixels (i_1, j_1) and (i_2, j_2), if $i_1 < i_2$ then $j_1 \leq j_2$. We begin by introducing a simple one-sided error algorithm that uses only sampling - it simply takes a sample of $\Theta(\widehat{w}^{1/2})$ 1-pixels and rejects if and only if it gets a pair that violates monotonicity. Clearly, a monotone matrix is never rejected, and we show that we are likely to encounter a pair of non-monotone 1-pixels in images that are far from monotone.

We next give an algorithm whose complexity is $\tilde{O}(n^{2/3}/(w^{1/3}))$, which improves on the simple sampling algorithm (in terms of the dependence on n) when $w(M) = \Omega(n^{4/5})$. This algorithm performs queries in addition to sampling. Roughly speaking, by sampling we try to detect violations that occur at relatively large distances, and by performing queries we try to detect violations that occur at relatively small distances.

The algorithm first takes a sample that with high probability either contains evidence that the matrix is not monotone (in a form of a "distant" pair of points that aren't monotone), or it (the sample) can be used to determine a set of submatrices with the following properties. First, the fraction of 1-pixels that reside outside the submatrices is relatively small. Second, there can be no violations between pairs of 1-pixels that reside in two different submatrices. Therefore, if the matrix is far from being monotone, then the violations are within the submatrices.

In the second stage of the algorithm we take an additional (small) sample, and for each sampled 1-pixel we perform queries within its submatrix (at varying distances from the selected 1-pixel) in order to detect violations with the sampled 1-pixels.

Acknowledgements: We wish to thank the FOCS 2010 reviewers for taking the time to read the article in depth, and for their many insightful comments.

REFERENCES

[1] N. Ailon and B. Chazelle. Information theory in property testing and monotonicity testing in higher dimensions. *Information and Computation*, 204:1704–1717, 2006.

[2] N. Alon and M. Krivelevich. Testing k-colorability. *SIAM Journal on Discrete Math*, 15(2):211–227, 2002.

[3] T. Batu, R. Rubinfeld, and P. White. Fast approximate PCPs for multidimensional bin-packing problems. *Information and Computation*, 196(1):42–56, 2005.

[4] A. Bhattacharyya, E. Grigorescu, K. Jung, S. Raskhodnikova, and D. Woodruff. Transitive-closure spanners. In *Proceedings of the 20th SODA*, pages 932–941, 2009.

[5] B. Chazelle, D. Liu, and A. Magen. Sublinear geometric algorithms. *SIAM Journal on Computing*, 35(3):627–646, 2006.

[6] A. Czumaj and C. Sohler. Property testing with geometric queries. In *Proceedings of the 9th ESA*, pages 266–277, 2001.

[7] A. Czumaj, C. Sohler, and M. Ziegler. Property testing in computation geometry. In *Proceedings of the 8th ESA*, pages 155–166, 2000.

[8] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. In *Proceedings of the 3rd RANDOM*, pages 97–108, 1999.

[9] F. Ergun, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000.

[10] E. Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1):107–116, 2004.

[11] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the 34th STOC*, pages 474–483, 2002.

[12] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

[13] 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.

[14] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, pages 302–343, 2002.

[15] S. Halevy and E. Kushilevitz. Distribution-free property testing. In *Proceedings of the 7th RANDOM*, pages 341–353, 2003.

[16] S. Halevy and E. Kushilevitz. Testing monotonicity over graph products. In *Proceedings of the 31st ICALP*, pages 721–732, 2004.

[17] T. Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004.

[18] I. Kleiner, D. Keren, and I. Newman. Applying property testing to an image partitioning problem. To appear in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2010.

[19] A. Matsliah, J. Briet, S. Chakraborty, and D. Garci'a-Soriano. Monotonicity testing and shortest-path routing on the cube. In *Proceedings of the 14th RANDOM*, 2010.

[20] M. Parnas and D. Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, 20(2):165–183, 2002.

[21] L. Rademacher and S. Vempala. Testing geometric convexity. In *Proceedings of the International Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 469–480, 2004.

[22] S. Raskhodnikova. Approximate testing of visual properties. In *Proceedings of the 7th RANDOM*, pages 370–381, 2003.

[23] D. Ron and G. Tsur. Testing properties of sparse images. Available from `www.eng.tau.ac.il~danar`, 2010.

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

[25] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, 17(2):264–280, 1971.