

# Testing *st*-Connectivity

Sourav Chakraborty<sup>1</sup>, Eldar Fischer<sup>2</sup>, Oded Lachish<sup>2</sup>, Arie Matsliah<sup>2</sup>,  
and Ilan Newman<sup>3,\*</sup>

<sup>1</sup> Department of Computer Science, University of Chicago, Chicago, IL-60637 USA  
sourav@cs.uchicago.edu

<sup>2</sup> Department of Computer Science, Technion, Haifa 3200, Israel  
{eldar,loded,ariem}@cs.technion.ac.il

<sup>3</sup> Department of Computer Science, University of Haifa, Haifa 31905, Israel  
ilan@cs.haifa.ac.il

**Abstract.** We continue the study, started in [9], of property testing of graphs in the orientation model. A major question which was left open in [9] is whether the property of *st*-connectivity can be tested with a constant number of queries. Here we answer this question on the affirmative. To this end we construct a non-trivial reduction of the *st*-connectivity problem to the problem of testing languages that are decidable by branching programs, which was solved in [11]. The reduction combines combinatorial arguments with a concentration type lemma that is proven for this purpose. Unlike many other property testing results, here the resulting testing algorithm is highly non-trivial itself, and not only its analysis.

## 1 Introduction

We continue the study, started in [9], of property testing of graphs in the orientation model. This is a model that combines information that has to be queried with information that is known in advance, and so does not readily yield to general techniques such as that of the regularity lemma used in [1] and [3].

Specifically, the information given in advance is an underlying undirected graph  $G = (V, E)$  (that may have parallel edges). The input is then constrained to be an orientation of  $G$ , and the distances are measured relative to  $|E|$  and not to any function of  $|V|$ . An orientation of  $G$  is simply an orientation of its edges. That is, for every edge  $e = \{u, v\}$  in  $E(G)$  an orientation of  $G$  specifies which of  $u$  and  $v$  is the source vertex of  $e$ , and which is the target vertex. Thus an orientation defines a directed graph  $\mathbf{G}$  whose undirected skeleton is  $G$ . Given the undirected graph  $G$ , a property of orientations is just a partial set of all orientations of  $G$ .

We study orientation properties in the framework of *property testing*. The meta problem in the area of property testing is the following: Given a combinatorial structure  $S$ , distinguish between the case that  $S$  satisfies some property  $\mathcal{P}$  and the case that  $S$  is  $\epsilon$ -far from satisfying  $\mathcal{P}$ . Roughly speaking, a combinatorial

---

\* Research supported in part by an Israel Science Foundation grant number 55/03.

structure is said to be  $\epsilon$ -far from satisfying some property  $\mathcal{P}$  if an  $\epsilon$ -fraction of its representation has to be modified in order to make  $S$  satisfy  $\mathcal{P}$ . The main goal in property testing is to design randomized algorithms, which look at a very small portion of the input, and use this information to distinguish with high probability between the above two cases. Such algorithms are called *property testers* or simply *testers* for the property  $\mathcal{P}$ . Preferably, a tester should look at a portion of the input whose size is a function of  $\epsilon$  only and is independent of the input size. A property that admits such a tester is called *testable*.

Blum, Luby and Rubinfeld [4] were the first to formulate a question of this type, and the general notion of property testing was first formulated by Rubinfeld and Sudan [14], who were interested in studying various algebraic properties such as the linearity of functions. The definitions and the first study of property testing as a framework were introduced in the seminal paper of Goldreich, Goldwasser and Ron [6]. Since then, an extensive amount of work has been done on various aspects of property testing as well as on studying particular properties. For comprehensive surveys see [13,5].

Here the relevant combinatorial structure is an orientation  $\mathbf{G}$  of the underlying graph  $G$ , and the distance between two orientations  $\mathbf{G}_1, \mathbf{G}_2$  is the number of edges that are oriented differently in  $\mathbf{G}_1$  and  $\mathbf{G}_2$ . Thus an orientation  $\mathbf{G}$  is  $\epsilon$ -far from a given property  $P$  if at least  $\epsilon|E(G)|$  edges have to be redirected in  $\mathbf{G}$  to make it satisfy  $P$ . Ideally the number of queries that the tester makes depends only on  $\epsilon$  and on nothing else (in particular it depends neither on  $|E|$  nor the specific undirected graph  $G$  itself).

A major question that has remained open in [9] is whether connectivity properties admit such a test. For a fixed  $s, t \in V(G)$ , an orientation  $\mathbf{G}$  is  $st$ -connected if there is a directed path from  $s$  to  $t$  in it. Connectivity and in particular  $st$ -connectivity is a very basic problem in graph theory which has been extensively studied in various models of computation.

Our main result is that the property of being  $st$ -connected is testable by a one-sided error algorithm with a number of queries depending only on  $\epsilon$ . That is, we construct a randomized algorithm such that for any underlying graph  $G$ , on input of an unknown orientation the algorithm queries only  $O(1)$  edges for their orientation and based on this decides with success probability  $\frac{2}{3}$  (this of course could be amplified to any number smaller than 1) between the case that the orientation is  $st$ -connected and the case that it is  $\epsilon$ -far from being  $st$ -connected. Our algorithm additionally has one-sided error, meaning that  $st$ -connected orientations are accepted with probability 1. Note that the algorithm knows the underlying graph  $G$  in advance and  $G$  is neither alterable nor part of the input to be queried. The dependence of the number of queries in our test on  $\epsilon$  is triply exponential, but it is independent of the size of the graph.

To put our result in context with previous works in the area of property testing, we note that graph properties were extensively studied since the early beginning in the defining paper of Goldreich, Goldwasser and Ron [6]. The model that was mainly studied is the *dense graphs* model in which an input is a graph represented as a subgraph of the complete graph. As such, for  $n$ -vertex graphs,

the input representation size is  $\binom{n}{2}$  which is the number of all possible unordered pairs. Thus, any property that has  $o(n^2)$  witness size, and in particular the property of undirected  $st$ -connectivity, is trivially testable as every input is close to the property. Similarly, properties of directed graphs were studied in the same context mostly by [1,3]. Inputs in this model are subgraphs of the complete directed graph (with or without anti parallel edges). In this case, directed  $st$ -connectivity is again trivial.

Other models in which graph properties were studied are the *bounded-degree* graph model of [7] in which a sparse representation of sparse graphs is considered (instead of the adjacency matrix as in the dense model), and the *general density* model (also called the *mixed* model) of [12] and [10]. In those models edges can be added as well, and so  $st$ -connectivity (directed or not) is again trivial as the single edge  $(s, t)$  can always be added and thus every graph is close to having the property. Testing general (all-pairs) connectivity is somewhat harder, and for this constant query testers are generally known, e.g. the one in [7] for the undirected bounded degree case.

Apart from [9], the most related work is that of [8] in which a graph  $G = (V, E)$  is given and the properties are properties of boolean functions  $f : E(G) \rightarrow \{0, 1\}$ . In [8] the interpretation of such a function is as an assignment to certain formulae that are associated with the underlying graph  $G$ , and in particular can be viewed as properties of orientations (although the results in [8] concentrate on properties that are somewhat more “local” than our “global” property of being  $st$ -connected). Hence, the results here should be viewed as moving along the lines of the investigation that was started in [9] and [8]. A common feature of the current work with this previous one, which distinguishes these results from results in many other areas of property testing and in particular the dense graph models, is that the algorithms themselves are rather non-trivial in construction, and not just in their analysis.

The algorithm that we present here for  $st$ -connectivity involves a preprocessing stage that is meant to reduce the problem to that of testing a branching program of bounded width. Once this is achieved, a randomized algorithm simulating the test for the constant width branching program from [11] is executed to conclude the result.

In general, the decision problem of  $st$ -connectivity of orientations of a given graph is not known to be reducible to constant width branching programs. In fact, it is most probably not the case as  $st$ -connectivity is complete for  $NL$  (non-deterministic LOG space) while deciding constant width branching programs is in  $L$ . In particular, it is not clear how to deal with high degree vertices or with large cuts. The purpose of the preprocessing stage is to get rid of these difficulties (here it will be crucial that we only want to distinguish between inputs that have the property and inputs that are quite far from having the property). This is done in several steps that constitute the main part of the paper. In particular we have an interim result in which most but not all edges of the graph are partitioned into constant width layers. This is proved using a weak concentration lemma for sequences of integers, which is formulated and proved for this purpose.

After the small portion of edges not in constant width layers is dealt with (using a reduction based on graph contraction), we can reduce the connectivity problem to a constant width read once branching program. Once such a branching program is obtained, the result of [11] can be used essentially in a black box manner.

Some interesting related open problems still remain. We still do not know if the property of being *strongly*  $st$ -connected is testable with a constant number of queries. The orientation  $\mathbf{G}$  is *strongly*  $st$ -connected if there is a directed path in  $\mathbf{G}$  from  $s$  to  $t$  as well as a directed path from  $t$  to  $s$ . A more general problem is whether in this model we can test the property of being *all-pairs strongly-connected* using a constant number of queries. Another related property is the property that for a given  $s \in V(G)$  every vertex is reachable by a directed path from  $s$ . The complexity of these problems is unknown, although there are some indications that similar methods as those used here may help in this regard.

The rest of this paper is organized as follows. In Section 2 we introduce the needed notations and definitions. Section 3 contains the statement of the main result and an overview of the proof. In Section 4 we reduce the problem of testing  $st$ -connectivity in general graphs to the problem of testing  $st$ -connectivity in nicely structured bounded-width graphs (we later call them *st-connectivity programs*). Due to space considerations, most of the proofs from Section 4 are omitted. In Section 5 we reduce from testing  $st$ -connectivity programs to testing *clustered* branching programs. Converting these clustered branching programs into regular ones, to which we can apply the testing algorithm from [11], is straightforward. Finally in Section 6 we combine these ingredients to wrap up the proof.

## 2 Preliminaries

### 2.1 Notations

In what follows, our graphs are going to possibly include parallel edges, so we use ‘named’ pairs for edges, i.e. we use  $e = e\{u, v\}$  for an undirected edge named  $e$  whose end points are  $u$  and  $v$ . Similarly, we use  $e = e(u, v)$  to denote the directed edge named  $e$  that is directed from  $u$  to  $v$ . Let  $G = (V, E)$  be an undirected multi-graph (parallel edges are allowed), and denote by  $n$  the number of vertices in  $V$ . We say that a directed graph  $\mathbf{G}$  is an *orientation* of the graph  $G$ , or in short a  $G$ -orientation, if we can derive  $\mathbf{G}$  from  $G$  by replacing every undirected edge  $e = e\{u, v\} \in E$  with either  $e(u, v)$  or  $e(v, u)$ , but not both. We also call  $G$  the *underlying graph* of  $\mathbf{G}$ .

Given an undirected graph  $G$  and a subset  $W \subset V$  of  $G$ ’s vertices, we denote by  $G(W)$  the induced subgraph of  $G$  on  $W$ , and we denote by  $E(W) = E(G(W))$  the edge set of  $G(W)$ . The distance between two vertices  $u, v$  in  $G$  is denoted by  $\text{dist}_G(u, v)$  and is set to be the length of the shortest path between  $u$  and  $v$ . Similarly, for a directed graph  $\mathbf{G}$ ,  $\text{dist}_{\mathbf{G}}(u, v)$  denotes the length of the shortest *directed* path from  $u$  to  $v$ . The distance of a vertex from itself is  $\text{dist}_{\mathbf{G}}(v, v) = \text{dist}_G(v, v) = 0$ . In the case where there is no directed path from  $u$  to  $v$  in  $\mathbf{G}$ ,

we set  $\text{dist}_G(u, v) = \infty$ . The diameter of an undirected graph  $G$  is defined as  $\text{diam}(G) = \max_{u, v \in V} \{\text{dist}_G(u, v)\}$ . Through this paper we always assume that the underlying graph  $G$  is connected, and therefore its diameter is finite.

For a graph  $G$  and a vertex  $v \in V$ , let  $\Gamma_{in}(v) = \{u : \exists e(u, v) \in E\}$  and  $\Gamma_{out}(v) = \{u : \exists e(v, u) \in E\}$  be the set of incoming and outgoing neighbors of  $v$  respectively, and let  $\Gamma(v) = \Gamma_{in}(v) \cup \Gamma_{out}(v)$  be the set of neighbors in the underlying graph  $G$ . Let  $\text{deg}_{in}(v)$ ,  $\text{deg}_{out}(v)$  and  $\text{deg}(v)$  denote the sizes of  $\Gamma_{in}(v)$ ,  $\Gamma_{out}(v)$  and  $\Gamma(v)$  respectively. We denote the  $i$ -neighborhood (in the underlying undirected graph  $G$ ) of a vertex  $v$  by  $N_i(v) = \{u : \text{dist}_G(u, v) \leq i\}$ . For example,  $N_1(v) = \{v\} \cup \Gamma(v)$ , and for all  $v \in V$ ,  $V = N_{\text{diam}(G)}(v)$ .

### 2.2 Orientation Distance, Properties and Testers

Given two  $G$ -orientations  $\mathbf{G}_1$  and  $\mathbf{G}_2$ , the distance between  $\mathbf{G}_1$  and  $\mathbf{G}_2$ , denoted by  $\Delta(\mathbf{G}_1, \mathbf{G}_2)$ , is the number of edges in  $E(G)$  having different directions in  $\mathbf{G}_1$  and  $\mathbf{G}_2$ .

Given a graph  $G$ , a property  $\mathcal{P}_G$  of  $G$ 's orientations is a subset of all possible  $G$ -orientations. We say that an orientation  $\mathbf{G}$  satisfies the property  $\mathcal{P}_G$  if  $\mathbf{G} \in \mathcal{P}_G$ . The distance of  $\mathbf{G}_1$  from the property  $\mathcal{P}_G$  is defined by  $\delta(\mathbf{G}_1, \mathcal{P}_G) = \min_{\mathbf{G}_2 \in \mathcal{P}_G} \frac{\Delta(\mathbf{G}_1, \mathbf{G}_2)}{|E(G)|}$ . We say that  $\mathbf{G}$  is  $\epsilon$ -far from  $\mathcal{P}_G$  if  $\delta(\mathbf{G}, \mathcal{P}_G) \geq \epsilon$ , and otherwise we say that  $\mathbf{G}$  is  $\epsilon$ -close to  $\mathcal{P}_G$ . We omit the subscript  $G$  when it is obvious from the context.

**Definition 1** ( $(\epsilon, q)$ -tester). *Let  $G$  be a fixed undirected graph and let  $P$  be a property of  $G$ 's orientations. An  $(\epsilon, q)$ -tester  $T$  for the property  $P$  is a randomized algorithm, that for any  $\mathbf{G}$  that is given via oracle access to the orientations of its edges operates as follows.*

- The algorithm  $T$  makes at most  $q$  orientation queries to  $\mathbf{G}$  (where on a query  $e \in E(G)$  it receives as the answer the orientation of  $e$  in  $\mathbf{G}$ ).
- If  $\mathbf{G} \in P$ , then  $T$  accepts it with probability 1 (here we define only one-sided error testers, since our testing algorithm will be one).
- If  $\mathbf{G}$  is  $\epsilon$ -far from  $P$ , then  $T$  rejects it with probability at least  $2/3$ .

The query complexity of an  $(\epsilon, q)$ -tester  $T$  is the maximal number of queries  $q$  that  $T$  makes on any input. We say that a property  $P$  is testable if for every  $\epsilon > 0$  it has an  $(\epsilon, q(\epsilon))$ -test, where  $q(\epsilon)$  is a function depending only on  $\epsilon$  (and independent of the graph size  $n$ ).

### 2.3 Connectivity Programs and Branching Programs

Our first sequence of reductions converts general  $st$ -connectivity instances to well structured bounded-width  $st$ -connectivity instances, as formalized in the next definition.

**Definition 2** ( $st$ -Connectivity Program). *An  $st$ -Connectivity Program of width  $w$  and length  $m$  over  $n$  vertices (or  $CP(w, m, n)$  in short), is a tuple*

$\langle G, \mathcal{L} \rangle$ , where  $G$  is an undirected graph with  $n$  vertices and  $\mathcal{L}$  is a partition of  $G$ 's vertices into layers  $L_0, \dots, L_m$ . There are two special vertices in  $G$ :  $s \in L_0$  and  $t \in L_m$ , and the edges of  $G$  are going only between vertices in consecutive layers, or between the vertices of the same layer, i.e. for each  $e = e\{u, v\} \in E(G)$  there exists  $i \in [m]$  such that  $u \in L_{i-1}$  and  $v \in L_i$ , or  $u, v \in L_i$ . The partition  $\mathcal{L}$  induces a partition  $E_1, \dots, E_m$  of  $E(G)$ , where  $E_i$  is the set of edges that have both vertices in  $L_i$ , or one vertex in  $L_{i-1}$  and another in  $L_i$ . In this partition of the edges the following holds:  $\max_i \{|E_i|\} \leq w$ .

Any orientation of  $G$ 's edges (that maps every edge  $e\{u, v\} \in E(G)$  to either  $e(u, v)$  or  $e(v, u)$ ) defines a directed graph  $\mathbf{G}$  in the natural way. An  $st$ -connectivity program  $C = \langle G, \mathcal{L}, \rangle$  defines a property  $P_C$  of  $G$ 's orientations in the following way:  $\mathbf{G} \in P_C$  if and only if in the directed graph  $\mathbf{G}$  there is a directed path from  $s$  to  $t$ .

Next we define branching programs. These are the objects to which we can apply the testing algorithm of [11].

**Definition 3 (Branching Program).** A Read Once Branching Program of width  $w$  over an input of  $n$  bits (or  $BP(w, n)$  in short), is a tuple  $\langle G, \mathcal{L}, X \rangle$ , where  $G$  is a directed graph with 0/1-labeled edges,  $\mathcal{L}$  is a partition of  $G$ 's vertices into layers  $L_0, \dots, L_n$  such that  $\max_i \{|L_i|\} \leq w$ , and  $X = \{x_0, \dots, x_{n-1}\}$  is a set of  $n$  Boolean variables. In the graph  $G$  there is one special vertex  $s$  belonging to  $L_0$ , and a subset  $T \subset L_n$  of accepting vertices. The edges of  $G$  are going only between vertices in consecutive layers, i.e. for each  $e = e(u, v) \in E(G)$  there is  $i \in [n]$  such that  $u \in L_{i-1}$  and  $v \in L_i$ . Each vertex in  $G$  has at most two outgoing edges, one of which is labeled by '0' and the other is labeled by '1'. In addition, all edges between two consecutive layers are associated with one distinct member of  $X = \{x_0, \dots, x_{n-1}\}$ . An assignment  $\sigma : X \rightarrow \{0, 1\}$  to  $X$  defines a subgraph  $G_\sigma$  of  $G$ , which has the same vertex set as  $G$ , and for every layer  $L_{i-1}$  (whose outgoing edges are associated with the variable  $x_{j_i}$ ), the subgraph  $G_\sigma$  has only the outgoing edges labeled by  $\sigma(x_{j_i})$ . A read once branching program  $B = \langle G, \mathcal{L}, X \rangle$  defines a property  $P_B \subset \{0, 1\}^n$  in the following way:  $\sigma \in P_B$  if and only if in the subgraph  $G_\sigma$  there is a directed path from the starting vertex  $s$  to any of the accepting vertices in  $T$ .

Branching programs that comply with the above definition can be tested by the algorithm of [11]. However, as we will see in Section 5, the branching programs resulting from the reduction from our  $st$ -connectivity programs have a feature that they require reading more than one bit at a time to move between layers. The next definition describes these special branching programs formally.

**Definition 4 (Clustered Branching Program).** A  $c$ -clustered Read Once Branching Program of width  $w$  and length  $m$  over an input of  $n$  bits (or shortly  $BP_c(w, m, n)$ ) is a tuple  $\langle G, \mathcal{L}, X, \mathcal{I} \rangle$ , where similarly to the previous definition,  $G$  is a directed graph with labeled edges (see below for the set of labels),  $\mathcal{L} = (L_0, \dots, L_m)$  is a partition of  $G$ 's vertices into  $m$  layers such that  $\max_i \{|L_i|\} \leq w$ , and  $X = \{x_0, \dots, x_{n-1}\}$  is a set of  $n$  Boolean variables. Here too,  $G$  has one

special vertex  $s$  belonging to  $L_0$ , and a subset  $T \subset L_n$  of accepting vertices. The additional element  $\mathcal{I}$  is a partition  $(I_1, \dots, I_m)$  of  $X$  into  $m$  components, such that  $\max_i \{|I_i|\} \leq c$ .

All edges in between two consecutive layers  $L_{i-1}$  and  $L_i$  are associated with the component  $I_i$  of  $\mathcal{I}$ . Each vertex in  $L_{i-1}$  has  $2^{|I_i|}$  outgoing edges, each of them labeled by a distinct  $\alpha \in \{0, 1\}^{|I_i|}$ .

An assignment  $\sigma : X \rightarrow \{0, 1\}$  to  $X$  defines a subgraph  $G_\sigma$  of  $G$ , which has the same vertex set as  $G$ , and for every layer  $L_i$  (whose outgoing edges are associated with the component  $I_i$ ), the subgraph  $G_\sigma$  has only the edges labeled by  $(\sigma(x_i))_{i \in I_i}$ . A  $c$ -clustered read once branching program  $B = \langle G, \mathcal{L}, X, \mathcal{I} \rangle$  defines a property  $P_B \subset \{0, 1\}^n$  in the following way:  $\sigma \in P_B$  if and only if in the subgraph  $G_\sigma$  there is a directed path from the starting vertex  $s$  to one of the accepting vertices in  $T$ .

Observe that  $BP(w, m)$  is equivalent to  $BP_1(w, m, m)$ .

### 3 The Main Result

For an undirected graph  $G$  and a pair  $s, t \in V(G)$  of distinct vertices, let  $P_G^{st}$  be a set of  $G$ -orientations under which there is a directed path from  $s$  to  $t$ . Formally,  $P_G^{st} = \{G : \text{dist}_G(s, t) < \infty\}$ .

**Theorem 1.** *The property  $P_G^{st}$  is testable. In particular, for any undirected graph  $G$ , two vertices  $s, t \in V(G)$  and every  $\epsilon > 0$ , there is an  $(\epsilon, q)$ -tester  $T$  for  $P_G^{st}$  with query complexity  $q = (2/\epsilon)^{2^{(1/\epsilon) \cdot 2^{O(\epsilon^{-2})}}}$ .*

Note that the property  $P_G^{st}$  is trivially testable whenever the undirected distance from  $s$  to  $t$  in  $G$  is less than  $\epsilon|E(G)|$ . In particular, our result is interesting only for sparse graphs, i.e. graphs for which  $|E(G)| \leq |V(G)|/\epsilon$ .

We can slightly improve the query complexity to  $q = (2/\epsilon)^{2^{O((1/\epsilon)^{(1/\epsilon))}}}$  by proving a stronger version of the concentration argument (Lemma 4), but we omit this proof from this extended abstract.

#### 3.1 Proof Overview

The main idea of the proof is to reduce the problem of testing  $st$ -connectivity in the orientation model to the problem of testing a Boolean function that is represented by a *small width read once branching program*. For the latter we have the result of [11] asserting that each such Boolean function is testable.

**Theorem 2 ([11]).** *Let  $P \subseteq \{0, 1\}^n$  be the language accepted by a read-once branching program of width  $w$ . Then testing  $P$  with one-sided error requires at most  $(\frac{2^w}{\epsilon})^{O(w)}$  queries.*

By the definition above of  $BP(w, n)$ , one could already notice that testing the acceptance of a branching program resembles testing  $st$ -connectivity, and that the two problems seem quite close. However, there are several significant differences, as noted here.

1. In branching programs, every layer is associated with a variable, querying which reveals all the edges going out from this layer. In  $st$ -connectivity instances, in order to discover the orientation of these edges we need to query each of them separately.
2. The length of the input in branching programs is the number of layers rather than the total number of edges.
3. The edges in branching program graphs are always directed from  $L_{i-1}$  to  $L_i$  for some  $i \in [n]$ . In our case, the graph is not layered, and a pair  $u, v$  of vertices might have any number of edges in both directions.
4. In branching programs the graphs have out-degree exactly 2, while an input graph of the  $st$ -connectivity problem might have vertices with unbounded out-degree.
5. The most significant difference is that the input graphs of the  $st$ -connectivity problem may have unbounded width. This means that the naive reduction to branching programs may result in an unbounded width  $BPs$ , which we cannot test with a constant number of queries.

We resolve these points in several steps. First, given an input graph  $G$ , we reduce it to a graph  $G^{(1)}$  which has the following property: for every induced subgraph  $W$  of  $G^{(1)}$ , the diameter of  $W$  is larger than  $\epsilon$  times the number of edges in  $W$ . Then we prove that  $G^{(1)}$  can be layered such that most of its edges lie within bounded-width layers. In particular, the total number of edges in the “wide” layers is bounded by  $\frac{\epsilon}{2}E(G^{(1)})$ . For this we need a concentration type lemma which is stated and proven here for this purpose. Next we reduce  $G^{(1)}$  to a graph  $G^{(2)}$ , which can be layered as above, but without wide layers at all. This in particular means that the number of edges in  $G^{(2)}$  is of the order of the number of layers, similarly to bounded width branching programs. In addition, in this layering of  $G^{(2)}$  the vertex  $t$  is in the last layer (while the vertex  $s$  remains in the first layer). Then we reduce  $G^{(2)}$  (which might be thought of as the *undirected* analogue of a bounded width branching program) to a clustered read once bounded width branching program. Finally we show that these clustered branching programs can be converted into non-clustered branching programs, to which we can apply the test from [11].

## 4 Reducing General Graphs to Connectivity Programs

In this section we make our first step towards proving Theorem 1. We reduce the problem of testing  $st$ -connectivity in a general graph to the problem of testing  $st$ -connectivity of an  $st$ -Connectivity Program. First we define the notion of reducibility in our context, and then we describe a sequence of reductions that will eventually lead us to the problem of testing read once bounded width  $BPs$ . Due to space considerations, some of the proofs from this section are omitted.

### 4.1 Reducibility Between $st$ -Connectivity Instances

Let  $\mathcal{G}^{st}$  denote the class of undirected graphs having two distinct vertices  $s$  and  $t$ , and let  $G, G' \in \mathcal{G}^{st}$ . We say that  $G$  is  $(\epsilon, \eta)$ -reducible to  $G'$  if there is a function  $\rho$  that maps orientations of  $G$  to orientations of  $G'$  (from now on we denote by  $\mathbf{G}'$  the orientation  $\rho(\mathbf{G})$ ) such that the following holds.

- If  $\mathbf{G} \in P_G^{st}$  then  $\mathbf{G}' \in P_{G'}^{st}$
- If  $\delta(\mathbf{G}, P_G^{st}) \geq \epsilon$  then  $\delta(\mathbf{G}', P_{G'}^{st}) \geq \eta$
- Any orientation query to  $\mathbf{G}'$  can be simulated by a single orientation query to  $\mathbf{G}$ .

We say that  $G$  is  $(\epsilon)$ -reducible to  $G'$  if it is  $(\epsilon, \epsilon)$ -reducible to  $G'$ . Notice that whenever  $G$  is  $(\epsilon, \eta)$ -reducible to  $G'$ , any  $(\eta, q)$ -tester  $T'$  for  $P_{G'}^{st}$  can be converted into an  $(\epsilon, q)$ -tester  $T$  for  $P_G^{st}$ . Or in other words,  $(\epsilon, q)$ -testing  $P_G^{st}$  is reducible to  $(\eta, q)$ -testing  $P_{G'}^{st}$ .

In the following section we introduce our first reduction, which is referred to as the reduction from  $G$  to  $G^{(1)}$  in the proof overview.

### 4.2 Reduction to Graphs Having High-Diameter Subgraphs

An undirected graph  $G$  is called  $\epsilon$ -long if for every subset of vertices  $W \subset V(G)$ ,  $\text{diam}(G(W)) > \epsilon|E(W)|$ .

**Lemma 1.** *Any graph  $G \in \mathcal{G}^{st}$  is  $(\epsilon)$ -reducible to a graph  $G' \in \mathcal{G}^{st}$  which is  $\epsilon$ -long.*

We first define a general contraction operator for graphs, and then use it for the proof. Given a graph  $G$  let  $W \subset V$  be a subset of its vertices, and let  $C_1, \dots, C_r$  be the vertex sets of the connected components of  $G(W)$ . Namely, for each  $C_i \subset W$ , the induced subgraph  $G(C_i)$  of the underlying graph  $G$  is connected, and for any pair  $u \in C_i, v \in C_j$  ( $i \neq j$ ) of vertices,  $e\{u, v\} \notin E(G)$ . We define a graph  $G/W$  as follows. The graph  $G/W$  has the vertex set  $V/W = (V \setminus W) \cup \{c_1, \dots, c_r\}$  and its edges are

$$E/W = \left\{ e\{u, v\} : (u, v \in V \setminus W) \wedge (e \in E) \right\} \cup \left\{ e\{c_i, v\} : (v \in V \setminus W) \wedge \exists u \in C_i (e\{u, v\} \in E) \right\}$$

Intuitively, in  $G/W$  we contract every connected component  $C_i$  of  $G(W)$  into a single (new) vertex  $c_i$  without changing the rest of the graph. Note that such a contraction might create parallel edges, but loops are removed. Whenever  $s \in C_i$  for some  $i \in [r]$ , we rename the vertex  $c_i$  by  $s$ , and similarly if  $t \in C_j$  then we rename  $c_j$  by  $t$ . In the following a connected component containing both  $s$  and  $t$  will not be contracted, so we can assume that the distinguished vertices  $s$  and  $t$  remain in the new graph  $G/W$ . Given an orientation  $\mathbf{G}$  of  $G$  we define the orientation  $\mathbf{G}/W = \rho(\mathbf{G})$  of  $G/W$  as the orientation induced from  $\mathbf{G}$  in the natural way (note that there are no “new” edges in  $G/W$ ).

**Lemma 2.** *Let  $W \subset V$  be a subset of  $G$ 's vertices, such that  $\text{diam}(G(W)) \leq \epsilon|E(W)|$ . Then  $G$  is  $(\epsilon)$ -reducible to the graph  $G/W$ .*

*Proof.* Fix an orientation  $\mathbf{G}$  of  $G$ . It is clear that if  $\mathbf{G} \in P_G^{st}$  then  $\mathbf{G}/W \in P_{G/W}^{st}$ . Now assume that  $\delta(\mathbf{G}, P_G^{st}) \geq \epsilon$ . Let  $d$  and  $d'$  denote  $\delta(\mathbf{G}, P_G^{st}) \cdot |E(G)|$  and  $\delta(\mathbf{G}/W, P_{G/W}^{st}) \cdot |E(G/W)|$  respectively. From the definition of the graph  $G/W$  it follows that  $d' \geq d - \text{diam}(G(W))$ . This is true since any  $st$ -path in  $\mathbf{G}/W$  can be extended to an  $st$ -path in  $\mathbf{G}$  by reorienting at most  $\text{diam}(G(W))$  edges in  $W$  (by definition,  $\text{diam}(G(W))$  is an upper bound on the undirected distance from any ‘‘entry’’ vertex to any ‘‘exit’’ vertex in  $G(W)$ ). From the condition on  $W$  we have  $|E(\mathbf{G}/W)| = |E| - |E(W)| \leq |E| - \frac{\text{diam}(G(W))}{\epsilon}$ . Combining these two together we have

$$\delta(\mathbf{G}/W, P_{G/W}^{st}) = \frac{d'}{|E(\mathbf{G}/W)|} \geq \frac{d - \text{diam}(G(W))}{|E| - \text{diam}(G(W))/\epsilon} \geq \frac{d - \text{diam}(G(W))}{d/\epsilon - \text{diam}(G(W))/\epsilon} = \epsilon$$

In addition, it is clear that we can simulate each query to  $\mathbf{G}/W$  by making at most one query to  $\mathbf{G}$ . □

Now the proof of Lemma 1 follows by applying this contraction (iteratively) for each ‘‘bad’’ subgraph  $W$ , until eventually we get a graph  $G'$  in which all vertex subsets  $W$  satisfy  $\text{diam}(G(W)) > \frac{|E(W)|}{\epsilon}$ . If in some stage we have both  $s$  and  $t$  contained in the contracted subgraph  $W$ , then we just output a tester that accepts all inputs (since in this case all orientations are  $\epsilon$ -close to being  $st$ -connected). Note that this process may result in several different graphs (depending on choices of the set  $W$  in each iteration), but we are only interested in one (arbitrary) graph  $G'$ . □

### 4.3 Properties of $\epsilon$ -Long Graphs

Next we show that an  $\epsilon$ -long graph  $G$  can be ‘‘layered’’ so that the total number of edges in the ‘‘wide’’ layers is bounded by  $\frac{\epsilon}{2}E(G)$ . We first define graph layering.

**Definition 5 (Graph layering and width).** *Given a graph  $G$  and a vertex  $s \in V(G)$ , let  $m$  denote the maximal (undirected) distance from  $s$  to any other vertex in  $G$ . We define a layering  $\mathcal{L} = (L_0, L_1, \dots, L_m)$  of  $G$ 's vertices as follows.  $L_0 = \{s\}$  and for every  $i > 0$ ,  $L_i = N_i(s) \setminus N_{i-1}(s)$ . Namely  $L_i$  is the set of vertices which are at (undirected) distance exactly  $i$  from  $s$ . Note that for every edge  $e\{u, v\}$  of  $G$  either both  $u$  and  $v$  are in the same layer, or  $u \in L_{i-1}$  and  $v \in L_i$  for some  $i \in [m]$ .*

We also denote by  $E_i^{\mathcal{L}}$  the subset of  $G$ 's edges that either have one vertex in  $L_i$  and the other in  $L_{i-1}$ , or edges that have both vertices in  $L_i$ . Alternatively,  $E_i^{\mathcal{L}} = E(L_i \cup L_{i-1}) \setminus E(L_{i-1})$ . We refer to the sets  $L_i$  and  $E_i^{\mathcal{L}}$  as vertex-layers and edge-layers respectively. We might omit the superscript  $\mathcal{L}$  from the edge-layers notation when it is clear from the context.

The vertex-width of a layering  $\mathcal{L}$  is  $\max_i\{|L_i|\}$ , and the edge-width of  $\mathcal{L}$  is  $\max_i\{|E_i^{\mathcal{L}}|\}$ .

**Bounding the Number of Edges Within Wide Edge-Layers.** The following lemma states that in a layering of an  $\epsilon$ -long graph most of the edges are captured in edge-layers of bounded width.

**Lemma 3.** *Consider the layering  $\mathcal{L}$  of an  $\epsilon$ -long graph  $G$  as defined above, and let  $I = \{i : |E_i| > 2^{100/\epsilon^2}/\epsilon\}$  be the set of indices of the wide edge-layers. Then the following holds:  $\sum_{i \in I} |E_i| \leq \frac{\epsilon}{2}|E|$ .*

*Proof.* Denote by  $\mathcal{A} = \langle a_0, a_1, \dots, a_m \rangle$  a sequence of integers, where  $a_0 = 1$  and for every  $i \geq 1$ ,  $a_i = |E_i|$ .

**Definition 6 ( $\epsilon$ -good sequence).** *Let  $0 < \epsilon < 1$  be a positive constant. A sequence  $1 = a_0, a_1, \dots, a_m$  of positive natural numbers is  $\epsilon$ -good if for every  $0 \leq k < m$  and  $\ell \in [m - k]$  we have that*

$$\sum_{i=k+1}^{k+\ell} a_i \leq \frac{\ell \cdot a_k}{\epsilon}.$$

**Claim 1.** *Let  $G$  be a graph in which for any induced subgraph  $W$  we have  $|E(W)| < \text{diam}(W)/\epsilon$ . Then the sequence  $\mathcal{A} = \langle a_0, a_1, \dots, a_m \rangle$  defined above is  $\epsilon/4$ -good.*

*Proof (Proof of Claim 1).* Let us assume the contrary of the claim. Let  $k$  and  $\ell$  be such that

$$\sum_{i=k+1}^{k+\ell} a_i > \frac{4\ell \cdot a_k}{\epsilon}$$

Consider the subgraph  $W$  defined by the vertices  $\cup_{i=k}^{k+\ell} L_i$  and the edges  $\cup_{i=k+1}^{k+\ell} E_i$ . Now the number of edges in  $W$  is clearly  $\sum_{i=k+1}^{k+\ell} a_i$ . For each vertex  $v$  in  $L_k$  consider the neighborhood of distance  $\ell + 1$  from  $v$ ,  $N_{\ell+1}(v)$ , and denote the subgraph it spans by  $W_v$ . Notice that each  $W_v$  is of diameter at most  $2(\ell + 1) \leq 4\ell$ , and that the union of the edge sets of the  $W_v$  includes the whole edge set of  $W$ , so we have  $\sum_{v \in L_k} |E(W_v)| \geq \sum_{i=k+1}^{k+\ell} a_i$ . The number of vertices in  $L_k$  is at most  $a_k$ , so by the pigeonhole principle we know that at least one of the vertices  $v$  in  $L_k$  has at least  $\frac{1}{a_k} \sum_{i=k+1}^{k+\ell} a_i$  edges in  $W_v$ . By our assumption on  $\sum_{i=k+1}^{k+\ell} a_i$  we have  $|E(W_v)| > \frac{4\ell}{\epsilon} \geq \text{diam}(W_v)/\epsilon$ , a contradiction.  $\square$

Proving Lemma 3 requires the following concentration type lemma, proof of which is omitted due to lack of space.

**Lemma 4 (The weak concentration lemma).** *Let  $\langle a_0, \dots, a_m \rangle$  be an  $\epsilon$ -good sequence and let  $B = \{i \mid a_i > 2^{5/\epsilon^2}/\epsilon\}$ . Then  $\sum_{i \in B} a_i \leq \epsilon \sum_{i=1}^m a_i$ .  $\square$*

Now the proof of Lemma 3 follows directly from Claim 1 and Lemma 4.  $\square$

#### 4.4 Reduction to Bounded Width Graphs

In this section we prove that  $\epsilon$ -long graphs can be reduced to graphs that have bounded width. In terms of the proof overview, we are going to reduce the graph  $G^{(1)}$  to the graph  $G^{(2)}$ .

Let  $G = (V, E) \in \mathcal{G}^{st}$ , and let  $\mathcal{L} = (L_0, L_1, \dots, L_m)$  be the layering of  $G$  as above. We call an edge-layer  $E_i$  *wide* if  $|E_i| > \frac{1}{\epsilon} \cdot 2^{100/\epsilon^2}$ . Let  $\mathcal{W}$  be the set of all wide edge-layers.

**Lemma 5.** *If  $G \in \mathcal{G}^{st}$  satisfies  $\sum_{E_i \in \mathcal{W}} |E_i| \leq \frac{\epsilon}{2} |E|$  then  $G$  is  $(\epsilon, \epsilon/2)$ -reducible to a graph  $G'$  which has no wide edge-layers at all.*

The proof of Lemma 5 appears in omitted.

#### 4.5 Reducing Bounded Width Graphs to $st$ -Connectivity Programs

So far we reduced the original graph  $G$  to a graph  $G'$  which has a layering  $\mathcal{L} = (L_0, L_1, \dots, L_m)$  of edge-width at most  $w = \frac{1}{\epsilon} \cdot 2^{100/\epsilon^2}$ , and in which the source vertex  $s$  belongs to layer  $L_0$ . The remaining difference between  $G'$  and an  $st$ -connectivity program is that in  $G'$  the target vertex  $t$  might not be in the last vertex-layer  $L_m$ . The following lemma states that we can overcome this difference by another reduction.

**Lemma 6.** *Let  $G'$  be a graph as described above. Then  $G'$  is  $(\epsilon, \epsilon/2)$ -reducible to an  $st$ -connectivity program  $S$  of width at most  $w + 1$ .  $\square$*

### 5 Reducing $st$ -Connectivity Programs to Branching Programs

We now show how to reduce an  $st$ -connectivity program to a clustered branching program (recall Definition 2 and Definition 4). First observe that we can assume without loss of generality that if an  $st$ -connectivity program has edge-width  $w$ , then its vertex-width is at most  $2w$  (since removing vertices of degree 0 essentially does not affect the  $st$ -connectivity program, and a vertex in  $L_i$  with edges only between it and  $L_{i+1}$  can be safely moved to  $L_{i+1}$ ).

Before moving to the formal description of the reduction, we start with a short intuitive overview. A branching program corresponds to a (space bounded) computation that moves from the start vertex  $s$ , which represents no information about the input at all, and proceeds (via the edges that are consistent with the input bits) along a path to one of the final vertices. Every vertex of the branching program represents some additional information gathered by reading more and more pieces of the input. Thus, the best way to understand the reduction is to understand the implicit meaning of each vertex in the constructed branching program.

Given a graph  $G$  of a bounded width  $st$ -connectivity program, and its layering  $L_0, L_1, \dots, L_m$ , we construct a graph  $G'$  (with layering  $L'_0, L'_1, \dots, L'_m$ ) for the bounded-width branching program. The graph  $G'$  has the same number of layers

as  $G$ . Each level  $L'_i$  in  $G'$  will represent the conditional connectivity of the vertices in the subgraph  $G_i = G(\bigcup_{j=0}^i L_j)$  of  $G$ . To be specific, the knowledge we want to store in a typical vertex at layer  $L'_i$  of  $G'$  is the following.

- for every  $u \in L_i$  whether it is reachable from  $s$  in  $G_i$ .
- for every  $v, u \in L_i$  whether  $v$  is reachable from  $u$  in  $G_i$ .

Hence, the amount of information we store in each node  $x \in L'_i$  has at most  $2w + (2w)^2$  many bits, and so there will be at most  $4^{2w^2+w}$  vertices in each  $L'_i$ , meaning that the graph  $G'$  of the branching program is of bounded width as well.

**Lemma 7.** *Let  $\epsilon > 0$  be a positive constant. Given a  $CP(w, m, n)$  instance  $C = \langle G, \mathcal{L} \rangle$ , we can construct a  $BP_w(4^{2w^2+w}, m, n)$  instance  $B = \langle G', \mathcal{L}', X', \mathcal{I}' \rangle$  and a mapping  $\rho$  from  $G$ -orientations to assignments on  $X$  such that the following holds,*

- if  $\mathbf{G}$  satisfies  $P_C$  then  $\sigma = \rho(\mathbf{G})$  satisfies  $P_B$ .
- if  $\mathbf{G}$  is  $\epsilon$ -far from satisfying  $P_C$  then  $\sigma = \rho(\mathbf{G})$  is  $\epsilon$ -far from satisfying  $P_B$ .
- any assignment query to  $\sigma$  can be simulated using a single orientation query to  $\mathbf{G}$ .

*Proof.* First we describe the construction, and then show that it satisfies the requirements above.

**The vertices of  $G'$ :** We fix  $i$  and show, based on the layer  $L_i$  of  $G$ , how to construct the corresponding layer  $L'_i$  of  $G'$ . Each vertex in  $L'_i$  corresponds to a possible value of a pair  $(S_i, R_i)$  of sets. The first set  $S_i \subseteq L_i$  contains vertices  $v \in L_i$  for which there is a directed path from  $s$  to  $v$  in the subgraph of  $G$  induced on  $\bigcup_{j=0}^i L_j$ . The second set  $R_i \subseteq L_i \times L_i$  is a set of ordered pairs of vertices, such that every ordered pair  $(u, v)$  is in  $R_i$  if there is a directed path from  $u$  to  $v$  in the subgraph of  $G$  induced on  $\bigcup_{j=0}^i L_j$  (the path can be of length 0, meaning that the  $R_i$ 's contain all pairs  $(v, v)$ ,  $v \in L_i$ ). Notice that  $|L'_i| = 2^{|L_i|^2 + |L_i|} \leq 4^{2w^2+w}$  for all  $i$ .

**The edges of  $G'$ :** Now we construct the edges of  $G'$ . Recall that  $E'_{i+1}$  denotes the set of (labeled) edges having one vertex in  $L'_{i+1}$  and the other in  $L'_i$ . Fix  $i$  and a vertex  $v \in L'_i$ . Let  $(S, R)$  be the pair of sets that correspond to  $v$ . Let  $S''$  be the set  $L_{i+1} \cap \Gamma_{out}(S)$ , namely the neighbors of vertices from  $S$  that are in  $L_{i+1}$ , and set  $R'' = \{(v, v) : v \in L_{i+1}\} \cup \{(u, v) : (u, v \in L_{i+1}) \wedge (v \in \Gamma_{out}(u))\} \cup \{(u, v) : (u, v \in L_{i+1}) \wedge ((\Gamma_{out}(u) \times \Gamma_{in}(v)) \cap R \neq \emptyset)\}$ . Now define  $R'$  as the transitive closure of  $R''$ , and set  $S' = S'' \cup \{v \in L_{i+1} : \exists u \in S'' (u, v) \in R'\}$ . Let  $v' \in L'_{i+1}$  be the vertex corresponding to the pair  $(S', R')$  that we defined above. Then the edges of  $E'_{i+1}$  are given by all such pairs of vertices  $(v, v')$ .

**The variables in  $X'$ :** Each variable  $x'_i \in X'$  is associated with an edge  $e_i \in E(G)$ . This association is actually the mapping  $\rho$  above, i.e. every orientation  $\mathbf{G}$  of  $G$  defines an assignment  $\sigma$  on  $X'$ .

**The partition  $\mathcal{I}'$  of  $X'$ :** Recall that  $E_i$  denotes the set of edges of  $G$  having either one vertex in  $L_{i-1}$  and the other in  $L_i$ , or both vertices in  $L_i$ . The partition  $\mathcal{I}'$  of  $X'$  is induced by the partition  $\mathcal{L}$  of  $V(G)$ . Namely, the component  $I_i$  of  $\mathcal{I}$  contains the set of variables in  $X'$  that are associated with edges in  $E_i$ . Thus  $w$  is also a bound on the sizes of the components in  $\mathcal{I}$ .

**The set  $T' \subset L'_m$  of accepting vertices:** The set  $T'$  is simply the subset of vertices in  $L'_m$  whose corresponding set  $S$  contains the target vertex  $t$  of  $G$ .

Note that each value of a variable in  $X'$  corresponds exactly to an orientation of an edge in  $G$ . This immediately implies the third assertion in the statement of the lemma. Distances between inputs are clearly preserved, so to prove the other assertions it is enough to show that the branching program accepts exactly those assignments that correspond to orientations accepted by the connectivity program. It is straightforward (and left to the reader) to see that the vertex reached in each  $L'_i$  indeed fits the description of the sets  $R$  and  $S$ , and so an assignment is accepted if and only if it corresponds to a connecting orientation.  $\square$

The branching programs resulting from the above reduction have a feature that they require reading more than one bit at a time to move between layers. Specifically, they conform to Definition 4. The result in [11], however, deals with standard branching programs (see Definition 3), which in relation to the above are a special case in which essentially  $m = n$  and all the  $I_i$ 's have size 1. Going from here to a standard (non-clustered) branching program (in which the edges between two layers depend on just one Boolean variable) is easy.

## 6 Wrapping Up – Proof of Theorem 1

We started with a graph  $G$  and wanted to construct an  $(\epsilon, q)$ -testing algorithm for  $st$ -connectivity of orientations of  $G$ . In Section 4.1, Section 4.2 and Section 4.3 we constructed a graph  $G_1$  such that if we have an  $(\epsilon, q)$ -test for  $st$ -connectivity in  $G_1$ , then we have an  $(\epsilon, q)$ -test for  $G$ . Additionally  $G_1$  has the property that most of the edge-layers in  $G_1$  are of size at most  $w = \frac{1}{\epsilon} \cdot 2^{100/\epsilon^2}$ . Then in Section 4.4 we constructed a graph  $G_2$  such that if we have an  $(\frac{\epsilon}{2}, q)$ -test for  $st$ -connectivity in  $G_2$  then we have an  $(\epsilon, q)$ -test for  $G_1$  and hence we have one for  $G$ . Moreover  $G_2$  has all its edge-layers of size at most  $w$ . Finally in Section 4.5 we built a graph  $G_3$  which has all its edge-layers of width at most  $w + 1$ , and in addition, the vertices  $s$  and  $t$  are in the first and the last vertex-layers of  $G_3$  respectively. We also showed that having an  $(\frac{\epsilon}{4}, q)$ -test for  $st$ -connectivity in  $G_3$  implies an  $(\frac{\epsilon}{2}, q)$ -test for  $G_2$ , and hence an  $(\epsilon, q)$ -test for  $G$ . This ends the first part of the proof, which reduces general graphs to  $st$ -connectivity programs.

Then in Section 5 from  $G_3$  we constructed a read once  $(w + 1)$ -clustered Branching Program that has width  $4^{2(w+1)^2+w+1}$  so that an  $(\frac{\epsilon}{4}, q)$ -test for this

$BP$  gives an  $(\frac{\epsilon}{4}, q)$ -test for  $st$ -connectivity in  $G_3$ . Then we converted the  $(w+1)$ -clustered Branching Program to a non-clustered Branching Program which has width  $w_1 = 4^{2(w+1)^2+(w+1)}2^{(w+1)}$ . Once we have our read once bounded width branching program then by applying the algorithm of [11] for testing branching programs we get an  $(\frac{\epsilon}{4}, q)$ -test with  $q = (\frac{2^{w_1}}{\epsilon/4})^{O(w_1)}$  queries for our problem. Hence by combining all of the above, we get an  $(\epsilon, q)$  testing algorithm for our original  $st$ -connectivity problem, where  $q = (2/\epsilon)^{2^{O((1/\epsilon)-2^{(100/\epsilon^2)})}}$ .  $\square$

## References

1. Alon, N., Fischer, E., Newman, I., Shapira, A.: A Combinatorial Characterization of the Testable Graph Properties: It's All About Regularity. In: Proceedings of the 38<sup>th</sup> ACM STOC, pp. 251–260. ACM, New York (2006)
2. Alon, N., Kaufman, T., Krivilevich, M., Ron, D.: Testing triangle-freeness in general graphs. In: Proceedings of the 17<sup>th</sup> SODA, pp. 279–288 (2006)
3. Alon, N., Shapira, A.: A Characterization of the (natural) Graph Properties Testable with One-Sided Error. In: Proceedings of the 46<sup>th</sup> IEEE FOCS, pp. 429–438 (2005) Also SIAM Journal on Computing (to appear)
4. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. Journal of Computer and System Sciences 47, 549–595 (1993) (a preliminary version appeared in Proc. 22<sup>nd</sup> STOC, 1990).
5. Fischer, E.: The art of uninformed decisions: A primer to property testing. In: Paun, G., Rozenberg, G., Salomaa, A. (eds.) Current Trends in Theoretical Computer Science: The Challenge of the New Century, vol. I, pp. 229–264. World Scientific Publishing (2004)
6. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. JACM 45(4), 653–750 (1998)
7. Goldreich, O., Ron, D.: Property testing in bounded degree graphs. Algorithmica 32(2), 302–343 (2002)
8. Halevy, S., Lachish, O., Newman, I., Tsur, D.: Testing Properties of Constraint-Graphs. In: Halevy, S., Lachish, O., Newman, I., Tsur, D. (eds.) Proceedings of the 22<sup>nd</sup> IEEE Annual Conference on Computational Complexity (CCC 2007) (to appear)
9. Halevy, S., Lachish, O., Newman, I., Tsur, D.: Testing Orientation Properties. In: Electronic Colloquium on Computational Complexity (ECCC) (2005)
10. Kaufman, T., Krivelevich, M., Ron, D.: Tight bounds for testing bipartiteness in general graphs. SICOMP 33(6), 1441–1483 (2004)
11. Newman, I.: Testing Membership in Languages that Have Small Width Branching Programs. SIAM Journal on Computing 31(5), 1557–1570 (2002)
12. Parnas, M., Ron, D.: Testing the diameter of graphs. Random Structures and Algorithms 20(2), 165–183 (2002)
13. Ron, D.: Property testing (a tutorial). In: Pardalos, P.M., Reif, J.H., Rolim, J.D.P. (eds.) Handbook of Randomized Computing, vol. II, ch. 15, Kluwer Press, Dordrecht (2001)
14. Rubinfeld, R., Sudan, M.: Robust characterization of polynomials with applications to program testing. SIAM Journal on Computing 25, 252–271 (1996) (first appeared as a technical report, Cornell University, 1993).