

Abstract

Property testing problems are relaxations of decision problems. A property testing algorithm (referred to as a *testing algorithm* or *tester*) has to decide if a given object has a prespecified property or is ϵ -far from the property (for a given distance parameter ϵ , and for a prespecified distance measure). The tester is given query access to the input, and is required to run in sublinear time.

In this thesis we focus on testing properties of directed graphs (digraphs). In particular we present the following results (where n is the number of vertices in the graph, d is the maximum degree, and d_{avg} is the average degree):

1. We present a testing algorithm for the property of Eulerianity in bounded-degree digraphs, which runs in time¹ $\tilde{O}(1/\epsilon)$. For unbounded-degree digraphs we show a lower bound of $\Omega(\sqrt{n/\epsilon})$, and give a testing algorithm that runs in time $\tilde{O}(\sqrt{n}/\epsilon^{3/2})$.
2. We consider the property of k -edge-connectivity in digraphs and present testing algorithms for both bounded-degree digraphs and unbounded-degree digraphs, that run in time $\tilde{O}((\frac{ck}{cd})^k d)$ and $\tilde{O}((\frac{ck}{cd_{\text{avg}}})^{k+1})$ respectively (where $c > 1$ is a constant). The same algorithm for unbounded-degree digraphs was given by Yoshida and Ito (to appear in *JSSC*), but our proof is more concise. In addition, we give a simpler analysis of the correctness of the testing algorithm for k -edge-connectivity in undirected graphs that was introduced by Goldreich and Ron (*Algorithmica*, 2002).
3. We study the property of k -vertex-connectivity, and give testing algorithms for both bounded-degree digraphs and unbounded-degree digraphs that run in time $\tilde{O}((\frac{ck}{cd})^k d)$ and $\tilde{O}((\frac{ck}{cd_{\text{avg}}})^{k+1})$ respectively. Moreover, we give a simpler analysis of the

¹The notation $\tilde{O}(g(k))$ stands for $O(g(k)) \cdot \text{polylog}(g(k))$.

testing algorithm of k -vertex-connectivity in undirected bounded-degree graphs that was shown by Yoshida and Ito (*ICALP 2008*) and extend the result to undirected unbounded-degree graphs.

4. We study the property of (k, ℓ) -edge-connectivity orientability in undirected graphs. We prove a lower bound of $\Omega(n)$ for a 1-sided-error testing algorithm.

Contents

1	Introduction	1
1.1	Background on testing graph properties	1
1.2	Our results	4
1.3	Organization	6
2	Preliminaries	8
3	Eulerianity	10
3.1	Testing Eulerianity in directed bounded-degree graphs	10
3.1.1	The testing algorithm	11
3.1.2	Correctness Proof of Algorithm 3.1	11
3.1.3	Query complexity and running time of Algorithm 3.1	16
3.1.4	Improving the running time of Algorithm 3.1	16
3.1.5	Correctness proof of Algorithm 3.2	17
3.1.6	Query complexity and running time of Algorithm 3.2	18
3.2	A lower bound on testing Eulerianity in unbounded-degree (sparse) digraphs	18
3.3	Testing Eulerianity in unbounded-degree digraphs	20
3.3.1	The testing algorithm	21
3.3.2	Correctness Proof of Algorithm 3.3	22
3.3.3	Query complexity and running time of Algorithm 3.3	25
4	k-edge-connectivity	26
4.1	Testing k -edge-connectivity in unbounded-degree digraphs	26
4.1.1	The testing algorithm	26

4.1.2	Correctness Proof of Algorithm 4.1	28
4.1.3	Query complexity and running time of Algorithm 4.1	30
4.2	Testing k -edge-connectivity in bounded-degree digraphs	31
4.3	Simplifying the correctness analysis of the testing algorithm for undirected k -edge-connectivity	32
5	k-vertex-connectivity	35
5.1	Testing k -vertex-connectivity in digraphs	35
5.1.1	The testing algorithm	35
5.1.2	Correctness proof of Algorithm 5.1	37
5.1.3	Query complexity and running time of Algorithm 5.1	40
5.2	Testing k -vertex-connectivity in bounded-degree digraphs	40
5.3	Reducing undirected k -vertex-connectivity to directed k -vertex-connectivity	43
5.4	Simplifying the proof for undirected k -vertex-connectivity	44
6	(k, ℓ)-edge-connectivity orientability	47
6.1	A characterization of the worst partition	49
6.2	A lower bound for 1-sided-error testing algorithms	51
7	Open problems	55
	Bibliography	56

List of Figures

3.1	An illustration for the matching of ‘ports’. Here u has one out-port, v has two out-ports, w has two in-ports and z has one in-port. The dashed edges are the edges of the perfect matching.	12
3.2	An illustration for the process of removing multiple edges and replacing them by other edges (while maintaining degree equalities). In part A of the figure, not all vertices other than u and v are necessarily distinct, but they are drawn in this manner for the sake of clarity. The dashed lines in parts B and C of the figure correspond to edges that are removed.	23
4.1	The BFS tree of a subset C , rooted at v . The bold edges are the cut edges, and each is removed in one of the recursive calls	30
6.1	The “bad” partition of graph G_{2k-1} . All subsets are single vertices	52
6.2	Partition F of graph G^+ . There are ℓ singleton subsets and one big subset S	53

List of Tables

1.1	Known results for testing properties of directed graphs depending on the edge density.	3
1.2	Summary of Eulerianity results	4
1.3	Summary of k -edge-connectivity results	5
1.4	Summary of k -vertex-connectivity results	6

Algorithms Listings

3.1	Testing Eulerianity in bounded-degree digraphs	11
3.2	Improved Testing Eulerianity in bounded-degree digraphs	17
3.3	Testing Eulerianity in unbounded-degree digraphs	22
4.1	Testing k -edge-connectivity in unbounded-degree digraphs	27
5.1	Testing k -vertex-connectivity in unbounded-degree digraphs	37

Chapter 1

Introduction

In this thesis we further the study of testing properties of directed graphs, which was first considered in [BR02]. We give testing algorithms for Eulerianity, for k -edge-connectivity and for k -vertex-connectivity in both bounded-degree digraphs and unbounded-degree digraphs. In addition, we prove a lower bound of a 1-sided-error testing algorithm of (k, ℓ) -edge-connectivity orientability, which is a property of undirected graphs.

In general, property testing problems [RS96] problems are relaxations of decision problems. A property tester has to decide whether a given object has a prespecified property or is far from having the property, with respect to some fixed distance measure between objects. That is, the property tester is given a distance parameter ϵ and is required to accept with high probability every object that has the property and to reject with high probability every object that is ϵ -far from having the property. In ‘ ϵ -far’ we mean that an ϵ -fraction of the object must be modified so that it obtains the property. To this end, the tester is given query access to the object, where the form of the queries depends on the type of object, and we are interested in testers that run in time *sublinear* in the size of the object. See [Gol98, Fis01, Ron01, Ron08, Ron10] for survey on property testing.

1.1 Background on testing graph properties

Testing properties of graphs has been extensively studied in the last few years. Several models have been studied, and we describe them below.

The adjacency matrix model (for dense graphs). This was the first model that was studied in the context of property testing [GGR98]. In this model a graph $G = (V, E)$ is represented by its $n \times n$ adjacency matrix (where $n = |V|$). The testing algorithm may query into the matrix (that is, it can ask whether there is an edge between u and v for any pair of vertices u and v). In this model, a graph is said to be ϵ -far from having a particular property if more than ϵn^2 edge modifications must be made so that it obtains the property. This model is appropriate for testing properties of dense graphs.

There are many graph properties for which there are efficient testers in this model. For example, graph partition properties (e.g. bipartiteness and k -colorability) are testable with query complexity $\text{poly}(1/\epsilon)$ [GGR98]. First-order graph properties (properties that can be formulated by first order expressions) are also testable in time independent of n [AFKS00]. A sequence of works by Alon and Shapira [AS05a, AS05c, AS05b], together with the work of Fischer and Newman [FN07], culminated in a characterization of all graph properties that are testable in the (undirected) dense-graphs model using a number of queries that is independent of n [AFNS06]. A different characterization, based on graph limits, was proved independently by Borgs et al. [BCL⁺06].

For directed dense graphs it is known that some of the property testers for undirected graphs (such as the tester for having a large cut) can be easily adapted to deal with the directed version of the property [GGR98]. In [AS04] there is a characterization of all (directed) subgraphs H for which H -freeness can be tested in time independent of n . Another basic property of directed graphs that was studied in the dense-graphs model is acyclicity. Acyclicity can be tested using $\tilde{O}(1/\epsilon^2)$ queries [BR02]. Note that in the undirected case of dense graphs, testing acyclicity is trivial since every dense graph is not acyclic.

The bounded-degree and the unbounded-degree graph models (for sparse graphs). In the bounded-degree model [GR02] graphs are represented by their incidence-lists, which have a bounded length d . The testing algorithm may probe the graphs incidence lists, that is, it may ask for the i th neighbor of vertex v , for any v of its choice. Here a graph is said to be ϵ -far from having the property if more than ϵdn edge modifications must be made so that the graph obtains the property. In the unbounded-degree model

there is no bound on the lengths of the lists. The algorithm is allowed the same type of queries as in the bounded-degree model, but the distance measure is different. Namely, a graph is said to be ϵ -far from having the property, if more than ϵm edge modifications must be made, where m is the number of edges in the graph (or a given upper bound on this number). Equivalently, the number of edge modification should be more than $\epsilon d_{\text{avg}} n$, where $d_{\text{avg}} = m/n$, that is half the average degree.

A variety of properties for undirected graphs are known to have efficient (independent of n) testers in the bounded-degree model and for some properties also in the unbounded-degree model. These include k -edge-connectivity (for $k \geq 1$), acyclicity and Eulerianity [GR02], and having a diameter of a bounded size [PR02], among others. In general, minor-closed properties in bounded-degree graphs are testable in time independent of n [BSS08, HKNO09]. There are other properties for which the testing algorithms have complexity that depends on n (sublinearly) such as bipartiteness [GR99] and (good) expansion [CS07, KS07, NS07] (where the dependence is almost optimal [GR02]).

Some of the testers for the aforementioned properties of undirected graphs can be adapted to deal with the directed case. This is true of (strong) connectivity [BR02] and of the diameter property [Izb04]. In particular, the query complexity and running time for both properties is $\text{poly}(1/\epsilon)$. These results hold only when it is possible to query both incoming and outgoing edges. On the other hand, if it is possible to query only outgoing edges (or only incoming edges), then there is a lower bound of $\Omega(\sqrt{n})$ for strong connectivity [BR02] (for a constant ϵ). Another basic property of directed graphs is *acyclicity*. While it is possible to test this property very efficiently in undirected graphs [GR02] and in directed graphs in the dense-graphs model [BR02], there is a lower bound of $\Omega(n^{1/3})$ for testing acyclicity of directed bounded-degree graphs [BR02].

The results stated above are summarized in Table 1.1

	dense	bounded	unbounded
Connectivity	trivial	in and out $\tilde{O}(1/\epsilon)$, out $\Omega(\sqrt{n})$	same as bounded
Diameter	trivial	–	$O(1/(\epsilon d_{\text{avg}})^3)$
Acyclicity	$\tilde{O}(1/\epsilon^2)$	$\Omega(n^{1/3})$	same as bounded

Table 1.1: Known results for testing properties of directed graphs depending on the edge density.

1.2 Our results

We now give more details on the properties studied in this thesis: Eulerianity, k -edge-connectivity, k -vertex-connectivity and (k, ℓ) -edge-connectivity orientability. For each property we give further context regarding related work.

Eulerianity. A graph is Eulerian if there exists a (closed) path that traverses each edge exactly once. Goldreich and Ron [GR02] presented a testing algorithm for Eulerianity in bounded-degree undirected graphs, which runs in time $\tilde{O}(1/\epsilon)$. The testing algorithm was extended to unbounded-degree graphs in [PR02] and the running time obtained is $\tilde{O}(1/(\epsilon d_{\text{avg}})^2)$. In this work we give a testing algorithm for the property of Eulerianity in directed graphs for both bounded-degree digraphs and unbounded-degree digraphs. The algorithm for bounded-degree digraphs runs in time $\tilde{O}(1/\epsilon)$, while the algorithm for unbounded-degree digraphs runs in time $\tilde{O}(\sqrt{n}/\epsilon^{3/2})$. The latter running time is almost optimal - we also present a lower bound of $\Omega(\sqrt{n}/\epsilon)$.

These results are valid when both incoming and outgoing edges can be queried. If only outgoing edges (or only incoming edges) can be queried, then for the bounded-degree case there is a lower bound of $\Omega(\sqrt{n})$ on the query complexity, which follows from the previously mentioned lower bound in [BR02] (on testing strong connectivity with only outgoing or incoming edges). For the unbounded-degree model, we observe that there is a lower bound of $\Omega(n)$ on the query complexity when only outgoing (or incoming) edges are allowed. The lower bound of $\Omega(\sqrt{n})$ also applies to testing directed k -edge-connectivity and k -vertex-connectivity, which are discussed next.

	Running Time	Reference
Bounded-degree Undirected	$\tilde{O}(1/\epsilon)$	[GR02]
Unbounded-degree Undirected	$\tilde{O}(1/(\epsilon d_{\text{avg}})^2)$	[PR02]
Bounded-degree Directed	$\tilde{O}(1/\epsilon)$	This work
Unbounded-degree Directed	$\tilde{O}(\sqrt{n}/\epsilon^{3/2}), \Omega(\sqrt{n}/\epsilon)$	This work

Table 1.2: Summary of Eulerianity results

k -edge-connectivity. A graph is k -edge-connected if there are k edge-disjoint paths from every vertex to any other vertex. A testing algorithm for k -edge-connectivity in

bounded-degree undirected graphs was given in [GR02]. Its running time is¹ $O\left(\frac{k^3 \log^2(1/\epsilon d)}{\epsilon^{3-\frac{2}{k}} d^{2-\frac{2}{k}}}\right)$. A similar algorithm was given for the unbounded case in [PR02] and its running time is $\tilde{O}(k^4/(\epsilon d_{\text{avg}})^4)$. An algorithm for testing strong-connectivity (that is, the case of $k = 1$ in directed graphs) for bounded-degree graphs was given by Bender and Ron [BR02] and its running time is $\tilde{O}(1/\epsilon d)$. We present a testing algorithm for any k , for both bounded-degree digraphs and unbounded-degree digraphs. The algorithm for bounded-degree digraphs runs in time $\tilde{O}((\frac{ck}{\epsilon d})^k d)$, while the algorithm for unbounded-degree digraphs runs in time $\tilde{O}((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1})$, for a constant $c > 1$. Independent from our work, Yoshida and Ito [YI09] gave a testing algorithm for k -edge-connectivity in bounded-degree digraphs with the same running time and query complexity.

	Running Time	Reference
Bounded-degree Undirected	$O\left(\frac{k^3 \log^2(1/\epsilon d)}{\epsilon^{3-\frac{2}{k}} d^{2-\frac{2}{k}}}\right)$	[GR02]
Unbounded-degree Undirected	$\tilde{O}(k^4/(\epsilon d_{\text{avg}})^4)$	[PR02]
Bounded-degree Directed	$\tilde{O}((\frac{ck}{\epsilon d})^k d)$	This work
Unbounded-degree Directed	$\tilde{O}((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1})$	This work

Table 1.3: Summary of k -edge-connectivity results

k -vertex-connectivity. A graph is k -vertex-connected if there are k vertex-disjoint paths from every vertex to any other vertex. A testing algorithm for k -vertex-connectivity in bounded-degree undirected graphs was given in [GR02] for $k = 1, 2, 3$. Its running time is $O(\text{poly}(1/\epsilon d))$. Yoshida and Ito [YI08] generalized the result and presented a testing algorithm for any k , that runs in time $\tilde{O}((\frac{ck}{\epsilon d})^k d)$. We present a simpler analysis of the correctness proof of their algorithm and show a testing algorithm for unbounded-degree undirected graphs. Its running time is $\tilde{O}((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1})$. In addition, we introduce testing algorithms for k -vertex-connectivity in bounded-degree digraphs and unbounded-degree digraphs. Their running times are $\tilde{O}((\frac{ck}{\epsilon d})^k d)$ and $\tilde{O}((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1})$ respectively, where $c > 1$ is a constant.

¹For the special cases of $k = 2$ and $k = 3$ there are algorithms with running-time $O\left(\frac{\log^2(1/\epsilon d)}{\epsilon}\right)$ and $O\left(\frac{\log^2(1/\epsilon d)}{\epsilon^2 d}\right)$, respectively [GR02].

	Running Time	Reference
Bounded-degree Undirected	$\tilde{O}\left(\left(\frac{ck}{\epsilon d}\right)^k d\right)$	[YI08]
Unbounded-degree Undirected	$\tilde{O}\left(\left(\frac{ck}{\epsilon d_{\text{avg}}}\right)^{k+1}\right)$	This work
Bounded-degree Directed	$\tilde{O}\left(\left(\frac{ck}{\epsilon d}\right)^k d\right)$	This work
Unbounded-degree Directed	$\tilde{O}\left(\left(\frac{ck}{\epsilon d_{\text{avg}}}\right)^{k+1}\right)$	This work

Table 1.4: Summary of k -vertex-connectivity results

(k, ℓ) -edge-connectivity orientability. This thesis concludes with the study of the property (k, ℓ) -edge-connectivity orientability. A directed graph is (k, ℓ) -edge-connected if there exists a vertex s , from which there are k edge-disjoint paths to any other vertex, and ℓ edge-disjoint paths from any other vertex to s . A graph is (k, ℓ) -edge-connectivity orientable if its edges can be oriented, so the oriented directed graph is (k, ℓ) -edge-connected. We consider the case of $\ell = 0$, and give a lower bound of $\Omega(n)$ on the complexity of any 1-sided-error testing algorithm for this property.

We note that the study of (k, ℓ) -edge-connectivity orientability does not fall within what is known as the *orientation model* [HLNT05]. In this model, an undirected graph is given to the algorithm in advance, and the input to be queried is a directed graph whose edges are orientations of the given undirected graph. Distances are measured with respect to the number of edges in the undirected graph, which determines how dense or sparse the problem is. In contrast to this model, when testing (k, ℓ) -edge-connectivity orientability, there is no fixed undirected graph that is given in advance, but rather there is query access to the (undirected) graph.

1.3 Organization

We first define the notations used throughout the thesis and define essential terms (Chapter 2). We then turn to presenting our results. In Chapter 3 we present testing algorithms for testing Eulerianity in bounded-degree digraphs and in unbounded-degree digraphs. We also prove a 2-sided-error lower bound for testing this property in unbounded-degree digraphs. In Chapter 4 we study the property of k -edge-connectivity. We give testing algorithms for both bounded-degree digraphs and unbounded-degree digraphs. We pro-

ceed to dealing with k -vertex-connectivity in Chapter 5 and present testing algorithms for both bounded-degree digraphs and unbounded-degree digraphs. Chapter 6 deals with the property of (k, ℓ) -edge-connectivity orientability in undirected graphs, where we give a lower bound for a 1-sided-error testing algorithm. Lastly, we discuss open questions in Chapter 7.

Chapter 2

Preliminaries

A graph $G = (V, E)$ is a set of vertices V and a set of edges E , where each edge is an unordered pair (u, v) , such that $u, v \in V$. We denote the number of vertices and edges by n and m , respectively (or possibly m is only a given upper bound on $|E|$). Unless stated otherwise, we allow multiple edges (so that E is a multiset of ordered pairs of vertices). The degree of vertex u is denoted by $d(u)$, and we let $d_{\text{avg}} \triangleq \frac{m}{n}$, so that d_{avg} is half the average degree in the graph. The degree of a set of vertices X is denoted by $d(X)$, which is the number of edges connecting a vertex from X and a vertex from $V \setminus X$. The set of edges connecting between one subset of vertices $C \subset V$ to another $D \subset V$ is denoted by $E(C, D)$. We let $\Gamma(v)$ denote the set of neighbors of v in undirected graphs.

A graph is directed if its edges are *ordered* pairs, meaning the edge (u, v) is an edge from vertex u to vertex v . The indegree of vertex u is denoted $d^+(u)$ and is the number of edges entering the vertex; that is, $d^+(u) = |\{(v, u) \in E\}|$. The outdegree of vertex u is denoted $d^-(u)$ and is the number of edges exiting the vertex; that is $d^-(u) = |\{(u, v) \in E\}|$. Similarly to $d(X)$ for a set of vertices X , we denote $d^+(X)$ the indegree of X and $d^-(X)$ the outdegree of X . In directed graphs we denote $\Gamma^+(v)$ the set of neighbors by incoming edges, while $\Gamma^-(v)$ denotes the set of neighbors by outgoing edges. In this work, whenever we use the term “graph” we mean a directed graph, unless noted otherwise. If we consider both undirected and directed graphs, then we use the term digraph for a directed graph.

We consider two models for testing graph properties. In the *bounded-degree* model, it is assumed that both the indegree and the outdegree of every vertex are bounded by a degree bound d , while in the *unbounded-degree* model, there is no such bound. In

both models it is possible to query “what is the other endpoint of the i th *outgoing* edge incident to v ” and “what is the other endpoint of the i th *incoming* edge incident to v ” for any choice of u and i . If u has less than i neighbors (in the queried direction), then a special symbol is returned. We refer to such queries as *neighbor queries*. As noted in the introduction, if only outgoing edges (or only incoming edges) can be queried, then there are strong lower bounds for the properties we study. We also assume that it is possible to query the indegree and outdegree of any vertex u . Note that if such queries are not allowed, then in the bounded-degree case each such query can be replaced by $\log d$ neighbor queries, while in the unbounded-degree case it can be replaced by $\log n$ neighbor queries.

For a graph property \mathcal{P} , in the bounded-degree model we say that a graph G is ϵ -far from having the property \mathcal{P} if the number of edge modification that should be performed so that the graph obtains the property is greater than ϵdn . In the unbounded-degree model we say that G is ϵ -far from having the property \mathcal{P} if the number of edge modifications that should be performed so that the graph obtains the property is greater than ϵm .

A property testing algorithm for a graph property \mathcal{P} is given a distance parameter ϵ and query access to a graph G . If G has the property \mathcal{P} then the algorithm accepts with probability at least $\frac{2}{3}$, and if G is ϵ -far from having the property \mathcal{P} , then the algorithm rejects with probability at least $\frac{2}{3}$. A *1-sided error* testing algorithm is a testing algorithm that accepts every G that has the property with probability 1.

We shall say that an event occurs with high constant probability if it occurs with probability $1 - \delta$ for some small constant δ . We assume that $\epsilon = \omega(1/n)$ or else we can query the whole graph and run an exact decision procedure.

Chapter 3

Eulerianity

3.1 Testing Eulerianity in directed bounded-degree graphs

A directed graph $G = (V, E)$ is *Eulerian* if there exists a directed cycle in the graph that traverses each edge in E exactly once. It is well known that a directed graph is Eulerian if and only if it is strongly connected and all vertices have indegree which is equal to the outdegree, more formally $\forall v \in V : d^+(v) = d^-(v)$.

An *Eulerian path* is a path that traverses each edge exactly once (as opposed to a circuit it does not have to end and start at the same vertex). A directed graph has an Eulerian path if and only if it is strongly connected and all vertices have indegree which is equal to the outdegree, except 2 vertices which may have a difference of at most one between their indegrees and their outdegrees. If there is such a pair of vertices, then one has a higher outdegree, while the other has a higher indegree.

A directed graph is strongly connected if and only if for every 2 vertices (u, v) there is a path from u to v and a path from v to u . For a graph G that is not strongly connected we consider the auxiliary graph defined by its *strong connectivity components (SCC)*. Each component is a maximal set of vertices, in which for each pair of vertices there is a path from one to the other. An SCC graph is a graph that represents the strongly connected components as single vertices and the edges are the edges between them (from the original graph). This graph is clearly a DAG (directed acyclic graph) and it must

have at least one *source* component (into which there are no incoming edges) and at least one *sink* component (out of which there are no outgoing edges). Source component are components, from which there are only outgoing edges, while for sink components there are only incoming edges.

3.1.1 The testing algorithm

The testing algorithm of Eulerianity in bounded-degree digraphs tests both strong connectivity and equality of the indegree and outdegree of vertices. That is, it tests two properties whose conjunction yields the desired property. This idea is similar to the testing algorithm of Eulerianity in undirected bounded-degree graphs presented in [GR02], but as we shall see, the analysis requires more care due to the difference between the property of Eulerianity in undirected and in directed graphs.

Algorithm 3.1 Testing Eulerianity in bounded-degree digraphs

TESTEULERIANITY(distance parameter ϵ , degree bound d):

1. Sample $s = \frac{16}{\epsilon d}$ vertices.
 2. From each sampled vertex perform a BFS twice - on outgoing edges and on incoming edges. Stop when $\frac{8}{\epsilon d}$ vertices have been reached on each direction or it is impossible to continue the search. If one of the searches reaches a dead-end, then REJECT.
 3. In addition, sample $k = \frac{24}{\epsilon}$ vertices.
 4. For each of the latter sampled vertices query their indegree and outdegree. If at least one vertex has unequal degrees, then REJECT.
 5. If no step caused rejection, then ACCEPT.
-

3.1.2 Correctness Proof of Algorithm 3.1

The tester combines two sub-testers, each checking a different property: degree equality and strong connectivity. Both properties are a sufficient and necessary condition for Eulerianity. However, the analysis does not directly reduce to showing that each of the two sub-testers is valid - as property testing of a conjunction of two sub-properties does not reduce in general to the property testing of each of the two sub-properties. Nonetheless, the following lemma does establish the validity of our tester.

Lemma 3.1. *Let G be a digraph that is ϵ -far from the class of directed Eulerian graphs with maximum degree d . Then, at least one of the following holds:*

1. G has more than $\frac{\epsilon}{12}n$ vertices which have unequal indegree and outdegree.
2. In G 's SCC graph the number of sink or source components is greater than $\frac{\epsilon}{4}dn$.

Proof. Assume, contrary to the lemma, that G has at most $\frac{\epsilon}{4}dn$ SCCs where each is either a sink or a source, and at most $\frac{\epsilon}{12}n$ vertices with unequal indegree and outdegree. We now show that by adding and removing at most ϵdn edges we can transform G into an Eulerian graph, contradicting the premise of the lemma that it is ϵ -far from Eulerianity.

First, we fix the unequal degrees by adding edges. To know which edges to add, we examine an auxiliary undirected bipartite graph $G' = (V = A \cup B, A \times B)$, where each node in subset A represents a missing outgoing edge in the original graph (we call them out-ports) and each node in subset B represents a missing incoming edge (we call them in-ports). Namely, for each vertex v such that $d^+(v) > d^-(v)$ there are $d^+(v) - d^-(v)$ nodes in A and for every v such that $d^-(v) > d^+(v)$, there are $d^-(v) - d^+(v)$ nodes in B . G' is a complete bipartite graph made of two parts: A and B .

We next show that there exists a perfect matching in G' . Such a matching determines what edges can be added in G so that in the indegrees and outdegrees of each vertex become equal.¹ For an illustration see Figure 3.1.2.

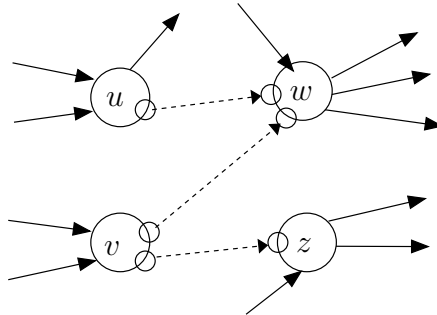


Figure 3.1: An illustration for the matching of ‘ports’. Here u has one out-port, v has two out-ports, w has two in-ports and z has one in-port. The dashed edges are the edges of the perfect matching.

¹In this process it is possible that multiple edges are created. If multiple edges are not allowed then there is a slightly different process, which is described in detail for the unbounded-degree case (where multiple edges are not allowed).

We look for a perfect matching in graph G' . Each edge (a, b) in the matching represents an added edge (u, v) in the original graph, where a represents a missing outgoing edge in u and b represents a missing incoming edge in v . We show that such a matching exists.

Since each edge contributes one to an outdegree and one to an indegree, we have that: $\sum_v d^+(v) = \sum_v d^-(v)$. Now, we separate the elements of the sums, summing separately the degrees of the edges with surplus outdegree, those with surplus indegree, and those with equal degrees:

$$\begin{aligned} & \sum_{v, d^+(v)=d^-(v)} d^+(v) + \sum_{v, d^+(v)>d^-(v)} d^+(v) + \sum_{v, d^+(v)<d^-(v)} d^+(v) \\ &= \sum_{v, d^+(v)=d^-(v)} d^-(v) + \sum_{v, d^+(v)>d^-(v)} d^-(v) + \sum_{v, d^+(v)<d^-(v)} d^-(v). \end{aligned} \quad (3.1)$$

By moving the sums around the two side of the equation:

$$\sum_{v, d^+(v)>d^-(v)} \left(d^+(v) - d^-(v) \right) = \sum_{v, d^-(v)>d^+(v)} \left(d^-(v) - d^+(v) \right). \quad (3.2)$$

The sum on the lefthand side of Equation (3.2) equals to $|A|$, while on the righthand side equals to $|B|$. Thus, we get that $|A| = |B|$, and the bipartite graph is $|A|$ -regular (all vertices in A are connected to all vertices in B), so there's a perfect matching. Thus, it is possible to fix the inequality by adding at most $|A|$ edges. $|A|$ equals to the sum of degree differences of vertices with higher outdegree. The number of those vertices is bounded by $\frac{\epsilon}{12}n$, since we assumed that there are at most $\frac{\epsilon}{12}n$ vertices with degree inequality. So, for fixing the graph's degree inequality, at most $\frac{\epsilon}{12}dn$ edge additions are necessary.

If the resulting graph is strongly connected then we are done. Otherwise, we show how to obtain a strongly connected graph while preserving the degree equalities and the degree bound d . To this end we examine the SCC graph. Note that before adding the edges to correct the degree inequalities, there were at most $\frac{\epsilon}{4}dn$ sink or source SCCs. Clearly, this remains true after adding any set of edges. We next claim that after adding the edges to make the indegrees and outdegree equal, every component is both a sink and a source component. That is, each SCC is isolated.

Claim 3.2. *Consider the SCC digraph of a digraph H in which $d_H^+(v) = d_H^-(v)$ for every vertex v . Then, every component in H is both a source component and a sink component.*

Proof. To verify this claim, assume first that there is a component that is a sink but is not a source. A similar argument holds for a source component that is not a sink. In such a case, the sum of the indegrees of its vertices is bigger than the sum of the outdegrees, and that contradicts the degree equality. But in such a case there can be no components that are neither sources nor sinks, and so each component is both a source component and a sink component. \square

Based Claim 3.2, we next show that in order to correct the strong connectivity of a graph with at most $\frac{\epsilon}{4}dn$ isolated components, at most $\frac{\epsilon}{2}dn$ edge modifications suffice. Combining this with the number of edges added to obtain degree equalities, we get a total of at most $\frac{\epsilon}{12}dn + \frac{\epsilon}{2}dn < \epsilon dn$ edge modifications, as required. Details follow.

Consider any order C_1, C_2, \dots, C_k of the isolated components in G . For each component C_i we do the following: if the component contains a vertex v such that $d^+(v) < d$ (so that $d^-(v) < d$ as well) then we add an incoming edge to v from C_{i-1} (in case of C_1 we connect from C_k) and an outgoing edge to C_{i+1} (in case of C_k we connect it to C_1). That's one edge modification per component. If all vertices in C_i have indegree and outdegree d , then we first remove one edge (u, v) from C_i (note that if the SCC is of size 1, then it is an isolated vertex, so the vertex has indegree and outdegree 0, which is smaller than d). We then connect an incoming edge from C_{i-1} to v and an outgoing edge going out from u to C_{i+1} . Note that the degree equality remains as well as the degree bound. Thus, at most 2 edge modifications are performed for each sink-and-source components (one removal and one addition).

In order to prove that the resulting graph is strongly connected, it suffices to show that for each pair (u, v) such that the edge from u to v was removed, there is a path from u to v in the (modified) graph. If after removing the edge (u, v) there is an additional path between u to v traversing the vertices of the strongly connected component, then we did not affect the connectivity. Otherwise, we show that there is now a new path from u to v in the modified graph using the added edges. Let us denote the pairs of edges we removed by $(u_1, v_1), \dots, (u_k, v_k)$ where u_i, v_i belong to an SCC C_i (if no edge

was removed, then let $u_i = v_i$ be the vertex to which an incoming and an outgoing edge were added). Since each C_i is an SCC, and furthermore is an isolated component, there is a path within each C_i from v_i to u_i , and this remains true after removing the edges (u_i, v_i) . Now, to reach v_i from u_i , we take the new edge (u_i, v_{i+1}) (if $i = k$, then the edge is (u_i, v_1)) and then the path (v_{i+1}, u_{i+1}) , and then the edge (u_{i+1}, v_{i+2}) and so on until v_i is reached.

Note that at any stage of the modification, the degree bound d was kept. Since for each vertex the higher degree (from the indegree and outdegree) was bounded by d in the original graph, then adding the edges as described above cannot violate the degree bound. In addition, when fixing the connectivity, both u and v preserved their degrees, so the bound is kept. This completes the proof of Lemma 3.1 \square

The next corollary follows from a simple counting argument:

Corollary 3.3. *If in G 's SCC graph the number of components which are sink or source components is greater than $\frac{\epsilon}{4}dn$, then G has at least $\frac{\epsilon}{8}dn$ sink or source components each containing less than $\frac{8}{\epsilon d}$ vertices.*

Proof. Assume, contrary to the claim, that in G 's SCC graph the number of components that are sink or source components is greater than $\frac{\epsilon}{4}dn$, and G has less than $\frac{\epsilon}{8}dn$ sink or source components each containing less than $\frac{8}{\epsilon d}$ vertices. This means that G has more than $\frac{\epsilon}{8}dn$ sink or source components each containing at least $\frac{8}{\epsilon d}$ vertices. It follows that G has more than $\frac{\epsilon}{8}dn \cdot \frac{8}{\epsilon d} = n$ vertices, contradicting the fact that $|V| = n$. \square

We are now ready to complete proving the correctness of Algorithm 3.1: Clearly, the algorithm always accepts an Eulerian graph. If the graph is ϵ -far from being Eulerian, then either it has at least $\frac{\epsilon}{8}dn$ source or sink components with less than $\frac{8}{\epsilon d}$ vertices or it has more than $\frac{\epsilon}{12}n$ vertices with unequal degrees. In the first case, the probability that none of the uniformly selected vertices belongs to such a component is at most $(1 - \frac{\epsilon d}{8})^s < \exp(-\frac{\epsilon d}{8}s)$, which is less than $1/3$ for $s = \frac{16}{\epsilon d}$. Given that at least one such vertex is selected, the algorithm rejects the graph as required. In the second case, the probability of not selecting such a vertex is $(1 - \frac{\epsilon}{12})^k < \exp(-k\frac{\epsilon}{12})$, which is less than $1/3$ for $k = \frac{24}{\epsilon}$.

3.1.3 Query complexity and running time of Algorithm 3.1

By the definition of the algorithm, its query complexity and running time are of the same order. In the first phase, the algorithm samples $\frac{16}{\epsilon d}$ vertices and performs a BFS of at most $\frac{8}{\epsilon d}$ vertices. The BFS running time is linear in the number of edges traversed during the search, which is at most $\frac{8}{\epsilon}$, since the degree bound is d . Therefore, the query complexity and running time of the first phase are bounded by $O(1/(\epsilon^2 d))$. In the second phase the algorithm samples $\frac{24}{\epsilon}$ vertices, and for each vertex sampled it performs 2 degree queries. It follows that the total query complexity and running time of the second phase are $\Theta(\frac{1}{\epsilon})$. Thus, the total complexity is $O(1/\epsilon \cdot \max(1, 1/(\epsilon d)))$. Note that testing Eulerianity in bounded-degree digraphs is relevant even if $\epsilon > \frac{1}{d}$. This is opposed to Eulerianity in undirected graphs, where at most n edge modification suffice to make any graph an Eulerian graph. Thus, we leave the term of $\max(1, 1/(\epsilon d))$ in the expression for the running time and query complexity of the algorithm.

3.1.4 Improving the running time of Algorithm 3.1

The query complexity and running time of the BFS executions can be improved to $O(\log(1/(\epsilon d))^2/\epsilon)$. This method of improvement was used in the testing algorithm for connectivity [GR02]. Roughly speaking, if many of the isolated small SCCs are “very small”, then on one hand, we need a bigger sample in order to “hit” one of them, but on the other hand, the BFS needs to traverse less edges. Alternatively, if the small SCCs are “not very small”, then the BFS needs to traverse more edges, but a smaller sample is sufficient to “hit” one of them. This is formalized in Algorithm 3.2 and its analysis.

Algorithm 3.2 Improved Testing Eulerianity in bounded-degree digraphs

IMPROVEDTESTEULERIANITY(distance parameter ϵ , degree bound d):

1. For $i = 1$ to $\ell = \log_2(8/(\epsilon d))$ do:
 - (a) Uniformly choose a set of $m_i = \frac{32 \cdot \log(8/(\epsilon d))}{2^i \epsilon d}$ vertices;
 - (b) For each vertex s chosen, perform a BFS starting from s until 2^i vertices have been reached or no new vertices can be reached. The BFS is performed twice - on incoming edges and on outgoing edges.
 2. If any of the above searches finds a small connected component, then output REJECT. Otherwise, move to checking degrees equality as in Algorithm 3.1.
 3. If didn't reject, then ACCEPT.
-

3.1.5 Correctness proof of Algorithm 3.2

Clearly, if G is Eulerian then it is accepted with probability 1. As for graphs that are far from the property:

Lemma 3.4. *If G is ϵ -far from the class of Eulerian graphs then the improved testing algorithm rejects it with probability at least $\frac{2}{3}$.*

Proof. Recall that by Lemma 3.1, since G is ϵ -far from the class of Eulerian graphs, at least one of the following holds: (1) G has more than $\frac{\epsilon}{12}n$ vertices which have unequal indegree and outdegree; (2) In G 's SCC graph the number of components that are either sink or source components is greater than $\frac{\epsilon}{4}dn$. In the first case, Algorithm 3.2 rejects with probability at least $2/3$, as was shown for Algorithm 3.1. It remains to deal with the latter case.

Let B_i be the set of sink or source components in G which contain at most $2^i - 1$ vertices and at least 2^{i-1} vertices. Recall that $\ell \triangleq \log_2(8/(\epsilon d))$. It follows from Corollary 3.3 that $\sum_{i=1}^{\ell} |B_i| \geq \frac{\epsilon}{8}dn$ (there are at least $\frac{\epsilon}{8}dn$ sink or source components of size at most $\frac{8}{\epsilon d}$). Hence, there exists an index $i \leq \ell$ such that $|B_i| \geq \frac{\epsilon}{8\ell}dn$. The number of vertices residing in components belonging to B_i is at least $2^{i-1} \cdot |B_i|$. So, the probability of choosing a vertex v , that belongs to one of these components is at least

$$\frac{2^{i-1} \cdot |B_i|}{n} \geq \frac{\epsilon d \cdot 2^i}{16\ell} = \frac{2}{s_i}. \quad (3.3)$$

It follows that, if the graph is far from Eulerianity due to having many small components, then with probability at least $\frac{2}{3}$, a vertex v belonging to a component in B_i is chosen in iteration i . The BFS starting from v discovers a small connected component leading to the rejection of G . \square

3.1.6 Query complexity and running time of Algorithm 3.2

The query complexity and running time of the BFS executions is bounded by

$$\sum_{i=1}^{\ell} s_i \cdot 2^i \cdot d = O\left(\frac{\log(1/\epsilon d)^2}{\epsilon}\right). \quad (3.4)$$

The query complexity and running time of checking degree equality is $O(\frac{1}{\epsilon})$. Hence the total query complexity and running time of the improved algorithm are $O(\max(1, \log(1/(\epsilon d))^2)/\epsilon)$.

3.2 A lower bound on testing Eulerianity in unbounded-degree (sparse) digraphs

In this section we show that there exists a lower bound of $\Omega(\sqrt{n/\epsilon})$ on the query complexity of any 2-sided-error Eulerianity testing algorithm for unbounded-degree digraphs. This lower bound holds even when the average degree is a constant.

In order to prove a lower bound for any 2-sided-error algorithm we introduce 2 graph families: one consists of Eulerian graphs, while the other one consists of graphs that are ϵ -far from Eulerian. We show that it is impossible to decide with high constant success probability whether a randomly selected graph from one of the the families, belongs to the first family or the second one by performing $o(\sqrt{n/\epsilon})$ queries. Each family is defined by a single underlying graph, and the graphs in the family differ only in the labeling of the vertices (where we consider all possible labelings).

Each graph in the first family (the Eulerian one) is a single directed cycle. This is clearly an Eulerian graph, and the average outdegree and indegree is 1. Each graph in the second family is composed of 2 subgraphs: one is a bipartite graph $G = (V = A \cup B, E)$

of $2\sqrt{2\epsilon n}$ vertices, where $|A| = |B| = \sqrt{2\epsilon n}$. There is exactly one edge between each pair of vertices (a, b) , where $a \in A$ and $b \in B$. The direction of all edges is from the vertices in part A to the vertices in part B . Clearly, every vertex has a large difference between its indegree and outdegree. The second subgraph is a cycle and it contains the rest of the vertices. These vertices have equal indegree and outdegree and are strongly connected. The 2 subgraphs are disconnected from each other. The average degree (either in or out) is $\frac{\sqrt{2\epsilon n} \cdot \sqrt{2\epsilon n} + (n - 2\sqrt{2\epsilon n})}{n}$, which is smaller than 2 for $\epsilon \leq \frac{1}{2}$.

Claim 3.5. *Every graph from the second family is ϵ -far from Eulerianity.*

Proof. In order to fix the Eulerianity of the second graph family we need to fix the inequality of the indegree and outdegree of the vertices in the bipartite subgraph and fix the connectivity between the 2 subgraphs. In order to fix the degree inequalities we must add or remove edges from each vertex of the bipartite subgraph. Since the absolute degree difference of each vertex is $\sqrt{2\epsilon n}$, we need at least $\frac{1}{2}\sqrt{2\epsilon n}$ edge modifications for each vertex. The reason is that each edge modification effects at most two vertices, so to fix degree differences of $2\sqrt{2\epsilon n}$ vertices, at least $\frac{2\sqrt{2\epsilon n} \cdot \sqrt{2\epsilon n}}{2} = 2\epsilon n$ edge modifications are necessary. In addition, to fix the connectivity it is enough to add 2 edges which is negligible, so we can disregard this addition. Hence, every graph of this family is ϵ -far from Eulerianity. \square

Lemma 3.6. *In order to distinguish with high constant success probability between a randomly selected graph in the first family and a randomly selected graph in the second family, it is necessary to perform $\Omega(\sqrt{n/\epsilon})$ queries.*

Proof. We analyze the number of queries needed in order to decide (with high probability) to which graph family a randomly selected graph (from either family) belongs. The main observation is that as long as a query does not “hit” the bipartite subgraph (in a graph from the second family), the distributions on the answers that the algorithm gets to its queries are identical for both families. This is true because each query on the cycle is answered by a uniformly selected vertex (as long as the cycle is not closed, which requires $\Omega(n)$ queries). Since there are $2\sqrt{2\epsilon n}$ vertices in the bipartite subgraph, if the algorithm performs less than $\sqrt{n/\epsilon}/c$ queries, then for a sufficiently large c , with high constant probability it will not obtain such a vertex. \square

This establishes the following lower bound:

Theorem 3.7. *In order to test the property of Eulerianity in unbounded-degree graphs, $\Omega(\sqrt{n/\epsilon})$ queries must be performed. The theorem holds even if the graph has a constant average degree.*

We observe that there is a lower bound of $\Omega(n)$ on testing Eulerianity in unbounded-degree digraphs, when the queries are on outgoing edges only (and similarly for incoming edges only). Namely, we show that any 2-sided-error testing algorithm that has query access to outgoing edges only must perform $\Omega(n)$ queries. To this end we introduce two graph families. The first family is the first family described above. Each graph in the second family is made up of $n - 1$ vertices in a directed cycle, and a unique vertex with edges going to all other vertices (and no incoming edges). Any graph from the second family is clearly $\frac{1}{2}$ -far from Eulerianity, since at least $n - 1$ edge modifications must be made and $d_{\text{avg}} = \frac{2(n-1)}{n}$. Obviously, to decide to which family a graph belongs to, the unique vertex must be sampled. All other vertices “look the same” when only outgoing edges can be queried. This establishes a lower bound of $\Omega(n)$ on the query complexity of any 2-sided-error algorithm for testing Eulerianity in unbounded-degree graphs, when only outgoing edges can be queried.

3.3 Testing Eulerianity in unbounded-degree digraphs

In this section we present an algorithm for testing Eulerianity in unbounded-degree digraphs. The algorithm is based on the fact that if a graph is far from being Eulerian, then it either has many edges that have at least one end-point whose outdegree does not equal its indegree, or it contains many small strongly connected components, which are a sink or a source component. The algorithm performs $O(\sqrt{n}/\epsilon^{3/2})$ queries, which is optimal in terms of the dependence on n (see Theorem 3.7). Here we consider only graphs with no multiple edges (or else the complexity of testing is $\Omega(n)$).²

²To verify this consider the following two classes of graphs. The first class consists of all labelings of a cycle over all vertices. In all graphs in the second class, all but two vertices are on a cycle, and between the remaining two vertices there are n uni-directional edges. Clearly every graph in the second family is $\Theta(1)$ -far from being Eulerian, but it is not possible to distinguish with sufficiently high constant probability between a randomly selected graph in one family and a randomly selected graph in the other family by performing less than n/c queries for some constant $c > 1$.

We introduce two new terms that will be used in the exposition:

1. A *biased vertex* is a vertex with unequal indegree and outdegree.
2. A *biased edge* is an edge such that one of its endpoints is a biased vertex.

3.3.1 The testing algorithm

In the first part of the testing algorithm, we try to find a biased edge. To this end, the algorithm runs a procedure for “almost-uniform edge sampling” [KKR04]. The procedure is given a parameter δ and it ensures that the probability that each edge is sampled is at least $1/(64 \cdot m)$ for a fraction of at least $(1 - \delta/4)$ of the edges. The number of queries performed and the running time are $\tilde{O}(\sqrt{n/\delta})$. This procedure was devised for undirected graphs, and we execute it on the undirected graph, denoted G' , that results from replacing each directed edge in G by an undirected edge. Since the algorithm has access both to incoming edges and to outgoing edges, it is possible to perform neighbor (and degree) queries in G' (by performing queries in G). The only problem that seems to arise is that if in G there is both a directed edge (u, v) and a directed edge (v, u) , then in G' we get two undirected edges between u and v . While the [KKR04] procedure was indeed designed and analyzed for graphs that have no multiple edges, it essentially works as is (with a constant factor increase in the complexity) if a constant edge multiplicity is allowed.

Algorithm 3.3 Testing Eulerianity in unbounded-degree digraphs

GENERALTESTEULERIANITY(distance parameter ϵ , average degree $d_{\text{avg}} = \frac{m}{n}$):

1. Sample $s = \frac{2048}{\epsilon}$ edges by running the [KKR04] procedure for almost-uniform edge sampling with the parameter δ set to $\epsilon/4$.
 2. For each sampled edge check if one of its endpoints is a biased vertex (by performing degree queries), implying that it is a biased edge.
 3. If a biased edge is found, then REJECT. Otherwise, proceed to the next phase.
 4. Sample $k = \frac{32}{\epsilon d_{\text{avg}}}$ vertices.
 5. From each sampled vertex perform a BFS twice - on outgoing edges and on incoming edges. Stop when $\frac{16}{\epsilon d_{\text{avg}}}$ vertices have been reached or it is impossible to continue the search. If one of the BFS executions reaches a dead-end, then REJECT.
 6. If no step caused rejection, then ACCEPT.
-

3.3.2 Correctness Proof of Algorithm 3.3

The next lemma establishes a condition (which is a disjunction of two conditions), that must hold if an unbounded-degree graph is ϵ -far from being Eulerian.

Lemma 3.8. *Let G be a digraph that is ϵ -far from the class of directed Eulerian graphs. Then at least one of the following two conditions holds:*

1. G has more than $\frac{\epsilon}{8}m$ biased edges.
2. In the SCC graph of G , the number of components that are sink or source components is greater than $\frac{\epsilon}{8}m$.

Proof. Assume, contrary to the claim, that the number of biased edges is at most $\frac{\epsilon}{8}m$ and that the number of sink or source components is at most $\frac{\epsilon}{8}m$. We shall show that such a graph can be made Eulerian by performing at most ϵm edge modifications, thus obtaining a contradiction to the fact the G is ϵ -far from Eulerianity. We first add edges to make the indegree and outdegree of each vertex equal. Since this might create multiple edges, we replace the multiple edges by other edges, in a manner that maintains degree equality (but may damage the connectivity of the graph). In the last step we make the graph connected. Details follow.

If the graph has at most $\frac{\epsilon}{8}m$ biased edges, then by definition we have that: $\frac{1}{2} \sum_{v \in V: v \text{ is biased}} |d^+(v) + d^-(v)| \leq \frac{\epsilon}{8}m$. As in the bounded-degree case, it is possible to perfectly match between the missing outgoing edges and the missing incoming edges for all biased vertices. Thus, by adding less than $\frac{\epsilon}{8}m$ edges we obtain a graph, denoted G' , in which there are no biased vertices.

We next describe how to remove all multiple edges in G' while maintaining the degree equalities. Consider a pair of vertices u, v such that there are $x > 1$ edges from u to v in G' . The idea is to replace all but one edge from u to v by pairs of edges of the form (u, w) and (w, v) . However, we need to perform such modification without creating new multiple edges, so some care is required, and it might be necessary to remove some additional edges, as described next.

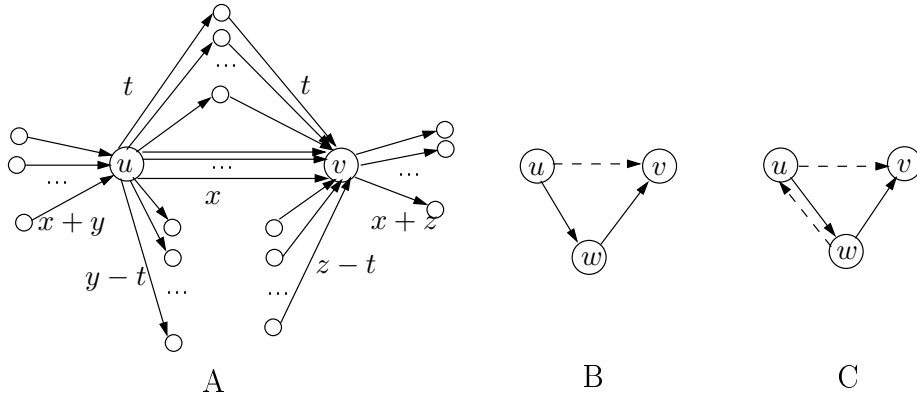


Figure 3.2: An illustration for the process of removing multiple edges and replacing them by other edges (while maintaining degree equalities). In part A of the figure, not all vertices other than u and v are necessarily distinct, but they are drawn in this manner for the sake of clarity. The dashed lines in parts B and C of the figure correspond to edges that are removed.

Let y denote the number of edges going from u to vertices other than v , let z denote the number of edges going to v from vertices other than u and let t denote the number of vertices that have both an (incoming) edge from u and an (outgoing) edge to v . For an illustration see Figure 3.2.A. As noted above, a simple “fix” would be to remove one edge from u to v and to add the edges (u, w) and (w, v) for some other vertex w . This preserves connectivity and degree equality, but might create new multiple edges. We first observe that if $x - 1 \leq (n - 2) - (y + z - t)$, then there are $x - 1$ vertices w such that $(u, w) \notin E(G')$ and $(w, v) \notin E(G')$, so that we may indeed replace each of the extra $x - 1$ edges from u to v by pairs of edges as described above. For an illustration see

Figure 3.2.B. Otherwise, $x - 1 > (n - 2) - (y + z - t)$. Assume, without loss of generality, that $y \geq z$, so that $x - 1 > (n - 2) - (2y - t)$, which implies that $x + 2y - t > n - 1$. Observe that $d_{G'}^+(u) = d_{G'}^-(u) = x + y$. This means that there is at least one vertex w , such that $(u, w), (w, u) \in E(G')$ while $(w, v) \notin E(G')$. We can now remove one multiple edge (u, v) , and replace the edge (w, u) by (w, v) (in the special case that $w = v$ we only remove (u, v) and $(w, u) = (v, u)$). This maintains the degree equalities of u, v and w , and reduces x by one. For an illustration see Figure 3.2.C. We may continue in this manner until x becomes 1 or $x - 1 \leq (n - 2) - (y + z - t)$ (in which case we proceed as described previously). Let the resulting graph (after applying this process to all multiple edges) be denoted by G'' .

Recall that we started with at most $\frac{\epsilon}{8}m$ SCCs. In the worst case, the removal of each edge increases the number of SCCs by one. Since the total number of (non-multiple) edges removed from G is upper bounded by the number of multiple edges in G' , which in turn is upper bounded by $\frac{\epsilon}{8}m$, the number of SCCs in G'' is at most $\frac{\epsilon}{4}m$. By Claim 3.2, all these SCCs are isolated components. Let's denote these components by (C_1, \dots, C_k) . Selecting a “representative” vertex from each C_i , and adding a directed cycle over the selected vertices, we obtain a strongly connected graphs while no new multiple edge is added (and no vertex becomes biased). Therefore, the resulting graph is Eulerian.

It remains to bound the number of edge modifications. The number of edges added in the preliminary (“matching”) process is at most $\frac{\epsilon}{8}m$, and hence the number of multiple edges created by this process is at most $\frac{\epsilon}{8}m$. The removal of each multiple edge is accompanied by at most two edge modifications (two additions or one addition and one removal). Lastly, to make the graph connected at most $\frac{\epsilon}{4}m$ edge additions are necessary. Therefore, the total number of edge modifications is at most $\frac{\epsilon}{8}m + 3 \cdot \frac{\epsilon}{8}m + \frac{\epsilon}{4}m < \epsilon m$, which contradicts the fact that the graph is ϵ -far from Eulerianity. \square

The next corollary follows from a simple counting argument (where the proof is similar to the proof of Corollary 3.3).

Corollary 3.9. *If a graph G has more than $\frac{\epsilon}{8}m$ sink or source components in its SCC graph, then G has at least $\frac{\epsilon}{16}m$ sink or source components each containing at most $\frac{16}{\epsilon d_{\text{avg}}}$ vertices.*

Now, for the correctness proof: if a digraph is Eulerian, it is always accepted, since it has no biased edges and it is connected. Otherwise, if the graph is ϵ -far from being Eulerian, then, by Lemma 3.8 it either has more than $\frac{\epsilon}{8}m$ biased edges, or, in its SCC digraph, the number of components that are sink or source components is greater than $\frac{\epsilon}{8}m$. In the former case, by sampling $\frac{2048}{\epsilon}$ edges “almost uniformly” using the [KKR04] procedure with the parameter δ set to $\frac{\epsilon}{4}$, we are ensured that with high constant probability, a biased edge will be selected with probability at least $1 - [(1 - 1/64) \cdot (\epsilon/16)]^{2048/\epsilon} > 2/3$, causing the algorithm to reject. In the latter case, by Corollary 3.9, G has at least $\frac{\epsilon}{16}m$ sink or source components each containing less than $\frac{16}{\epsilon d_{\text{avg}}}$ vertices. Therefore, with high constant probability, a vertex from such a small SCC is selected in the second stage of the algorithm, causing the algorithm to reject.

3.3.3 Query complexity and running time of Algorithm 3.3

As in the bounded-degree case, the running time and query complexity are of the same order. Sampling one edge by the [KKR04] procedure takes $\tilde{O}(\sqrt{n}/\epsilon)$ time. The running time of the BFS is $O(1/(\epsilon d_{\text{avg}})^2)$ (the BFS running time is linear in the number of edges it traverses). The improved version of the BFS takes $O(\log(1/(\epsilon d_{\text{avg}}))^2/\epsilon)$. Thus, the total running time of the algorithm is $\tilde{O}(\sqrt{n}/\epsilon^{3/2} + \log(1/(\epsilon d_{\text{avg}}))^2/\epsilon) = \tilde{O}(\sqrt{n}/\epsilon^{3/2})$.

Chapter 4

k-edge-connectivity

A graph $G = (V, E)$ is *k*-edge-connected if for every (ordered) pair of vertices (v, u) there are *k* edge-disjoint paths from v to u . Equivalently, G is *k*-edge-connected, if for every subset S , we have that $d^+(S) \geq k$. A testing algorithm for the analogous property of testing *k*-edge-connectivity in undirected graphs was given in [GR02]. Independently from our work, Yoshida and Ito [YI09] presented an algorithm for testing *k*-edge-connectivity in bounded-degree digraphs. Their algorithm is the same algorithm as ours, but our analysis is different. Here we present a testing algorithm for both the bounded-degree case and the unbounded-degree case. We start by considering the unbounded-degree case.

4.1 Testing *k*-edge-connectivity in unbounded-degree digraphs

4.1.1 The testing algorithm

Roughly speaking, similarly to the algorithm for testing *k*-edge-connectivity in undirected (bounded-degree) graphs [GR02], the testing algorithm for the directed case builds on the fact that a graph that is far from being *k*-edge-connected has “many” subsets that are “small” and have a (directed) edge-cut smaller than *k* (of either incoming edges or outgoing edges). The algorithm tries to find at least one such subset, which provides evidence that the graph is not *k*-edge-connected.

The main building block of the algorithm is a recursive procedure whose input is a

vertex v , an upper bound, denoted ℓ , on the number of vertices that should be reached, an upper bound on the size of the edge-cut, denoted t , the direction ('out', i.e. '-' or 'in', i.e. '+') to work on, denoted σ , and a set F of *forbidden* edges. The procedure is initially called with $t = k - 1$ and $F = \emptyset$. The procedure determines if v belongs to a subset S , such that: (1) the size of S is at most ℓ ; (2) $d_{(V, E \setminus F)}^\sigma(S) \leq t$, where $d_{(V, E \setminus F)}^\sigma(S)$ denotes $d^\sigma(S)$ in the graph $(V, E \setminus F)$. The procedure returns TRUE if and only if the vertex belongs to such a subset. Otherwise, it returns FALSE. The procedure works by running a BFS recursively. In each level of the recursion it removes a single edge in the BFS tree, and calls itself recursively with an upper bound of $t - 1$ on the size of the edge-cut. This is a variant of the procedure "ExhaustSearch" presented in [YI08].

Procedure 4.1.: Deciding if a vertex belongs to a small set with a small edge-cut (input: v, σ, ℓ, t, F)

If $\sigma = +$ perform the following on incoming edges, otherwise on outgoing edges:

1. *Run a BFS from v with the restriction that no edge in F can be traversed, until $(\ell + 1)$ vertices have been reached. Let X be the set of edges in the BFS tree.*
2. *If the BFS reached a dead-end before reaching $\ell + 1$ vertices, then return TRUE.*
3. *If $t = 0$, then return FALSE.*
4. *For each edge $e \in X$ run Procedure 4.1 with parameters $v, \sigma, \ell, t - 1$ and $F \cup \{e\}$. If any execution returns TRUE, then return TRUE. Otherwise, return FALSE.*

Algorithm 4.1 Testing k -edge-connectivity in unbounded-degree digraphs

GENERALTESTKEDGECONN(distance parameter ϵ , average degree $d_{\text{avg}} = \frac{m}{n}$):

1. Sample $s = \Theta\left(\frac{k}{\epsilon d_{\text{avg}}}\right)$ vertices uniformly and independently.
 2. For each sampled vertex v run Procedure 4.1 with parameters $v, \ell = \frac{2k}{\epsilon d_{\text{avg}}}, t = k - 1, F = \emptyset$ twice: once with $\sigma = -$ and once with $\sigma = +$.
 3. If one of the executions of Procedure 4.1 returns TRUE, then REJECT.
 4. If no step caused rejection, then ACCEPT.
-

4.1.2 Correctness Proof of Algorithm 4.1

We start by quoting a theorem of Frank [Fra92] on which our analysis is based:

Theorem 4.2 ([Fra92]). *A directed graph $G = (V, E)$ can be made k -edge-connected (for $k \geq 1$) by adding at most m^* (directed) edges if and only if $\sum_i (k - d^+(X_i)) \leq m^*$ and $\sum_i (k - d^-(X_i)) \leq m^*$ hold for every family of disjoint subsets $\{X_1, \dots, X_t\}$ of vertices.*

By setting m^* to ϵm we get a necessary and sufficient condition for a graph being ϵ -close to k -edge-connectivity. By negating the condition we get a necessary and sufficient condition for being ϵ -far from the property of k -edge-connectivity:

Corollary 4.3. *A directed graph $G = (V, E)$ is ϵ -far from being k -edge-connected (for $k \geq 1$) if and only if there exists a family of disjoint subsets $\{X_1, \dots, X_t\}$ of vertices for which either $\sum_i (k - d^+(X_i)) > \epsilon m$ or $\sum_i (k - d^-(X_i)) > \epsilon m$.*

We next show that if a graph is far from being k -edge-connected, then it has “many” subsets that are “small” and for which the number of outgoing or incoming edges is less than k .

Lemma 4.4. *In a graph G that is ϵ -far from k -edge-connectivity, there are at least $\frac{\epsilon m}{2k}$ disjoint subsets, each of size at most $\frac{2k}{\epsilon d_{\text{avg}}}$, with an incoming edge-cut or an outgoing edge-cut of size at most $k - 1$.*

Proof. Let $G = (V, E)$ be a graph that is ϵ -far from being k -edge-connected, By Corollary 4.3 there exists a partition $\{X_1, \dots, X_t\}$ for which $\sum_i (k - d^+(X_i)) > \epsilon m$ or $\sum_i (k - d^-(X_i)) > \epsilon m$. Assume that the former holds (the analysis of the latter case is analogous). Since $d^+(X_i) \geq 0$ for every X_i , the maximal value of each term $(k - d^+(X_i))$ in the sum is k . Let $\{X_{i_1}, \dots, X_{i_{t'}}\}$ be a subpartition of t' subsets for which $d^+(X_i) < k$ (we ignore subsets for which $d^+(X_i) \geq k$ since they don't contribute a positive value to the sum). It follows that $kt' > \sum_i (k - d^+(X_i)) > \epsilon m$, so that $t' > \frac{\epsilon m}{k}$. By a simple counting argument (similar to one we have applied before in the proof of Corollary 3.3) we get that there are at least $\frac{\epsilon m}{2k}$ disjoint subsets, each of size at most $\frac{2k}{\epsilon d_{\text{avg}}}$, with an edge-cut (either incoming or outgoing) of size strictly smaller than k . \square

One additional claim regarding vertices that belong to subsets with a bounded-size edge-cut is needed. The claim establishes that by traversing the vertices reachable from v we can find a bounded-size edge-cut.

Claim 4.5. *If a vertex v belongs to a subset C for which $d^-(C) < k$, then there exists a subset $C' \subseteq C$ such that v can reach any vertex in C' and $d^-(C') < k$. Analogously, if a vertex v belongs to a subset C for which $d^+(C) < k$, then there exists a subset $C' \subseteq C$ such that any vertex in C' can reach v and $d^+(C') < k$.*

Proof. We prove the claim for the case of $d^-(C) < k$. The proof for the case of $d^+(C) < k$ is analogous. Let $C' \subseteq C$ consist of all vertices in C that can be reached from v . If $C' = C$ then the claim holds by the premise that $d^-(C) < k$. Otherwise, by the definition of C' , there are no edges going from vertices in C' to vertices in $C \setminus C'$. Therefore, the only outgoing edges incident to vertices in C' are to vertices in $V \setminus C$. Therefore, $d^-(C') \leq d^-(C) < k$, as claimed. \square

The Correctness of Algorithm 4.1. Before proving the correctness of the testing algorithm, we prove the correctness of Procedure 4.1:

Lemma 4.6. *Suppose that Procedure 4.1 is given a vertex v that can reach, in the direction indicated by σ and without traversing any edge in F , a set of vertices C' such that $|C'| \leq \ell$ and $d_{(V, E \setminus F)}^\sigma(C') \leq t$. Then the procedure returns TRUE.*

Proof. We prove the lemma by induction on $r = d_{(V, E \setminus F)}^\sigma(C')$. The base of induction: $r = 0$, so that the BFS surely reaches a dead-end before reaching $\ell + 1$ vertices (since $|C'| \leq \ell$), and TRUE is returned. The induction step: we prove the claim for $r > 0$, based on the induction hypothesis that the claim holds for $r - 1 \geq 0$.

The BFS runs until it reaches $\ell + 1$ vertices or it reaches a dead-end. In the latter case TRUE is returned, and it remains to deal with the former case. Let Y denote the set of vertices reached by the BFS. Since $|C'| \leq \ell$, we have that Y contains at least one vertex, denoted y , that does not belong to C' . In order to reach y , necessarily, one of the edges $e \in E(C', V \setminus C') \setminus F$ had to be traversed, and thus e belongs to the BFS tree. The procedure calls itself ℓ times, each time removing a different edge from the BFS tree (i.e., adding the edge to F in the recursive call), and reducing the bound on the size of the

edge-cut. This ensures that in one of those calls, an edge that belongs to $E(C', V \setminus C') \setminus F$ is removed (added to F). In this call the procedure is given v , that can reach (in the direction σ , and without traversing any edge in F) a subset C' of size at most $\ell - 1$, such that $d_{(V, E \setminus F)}^\sigma(C') = r - 1$. For this call the induction hypothesis holds, and thus `TRUE` is returned. The algorithm returns `TRUE` if at least one of the calls returned `TRUE`, and so `TRUE` is returned. \square

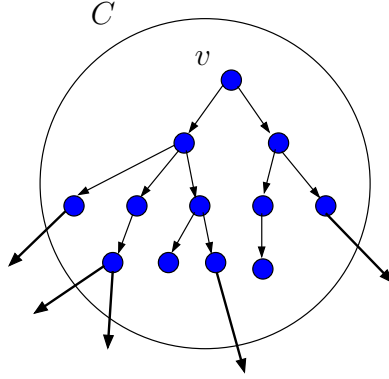


Figure 4.1: The BFS tree of a subset C , rooted at v . The bold edges are the cut edges, and each is removed in one of the recursive calls

It is clear that the algorithm accepts any k -edge-connected graph, since every subset of vertices has an edge-cut of size at least k (of both incoming and outgoing edges). Thus, the BFS executions (for every F they are called with, since $|F| \leq k - 1$) can always reach more than $\frac{2k}{\epsilon d_{\text{avg}}}$ vertices. If the graph is ϵ -far from being k -edge-connected, then at least one vertex v from a subset X_i for which $|X_i| \leq \frac{2k}{\epsilon d_{\text{avg}}}$ and either $d^+(X_i) < k$ or $d^-(X_i) < k$ is sampled with high constant probability. Conditioned on this event, we assume without loss of generality that $d^-(X_i) < k$. By Claim 4.5, the vertex v can reach a subset $C' \subseteq X_i$ such that $d^-(C') < k$ (and clearly $|C'| \leq |X_i| \leq \frac{2k}{\epsilon d_{\text{avg}}}$). Since Procedure 4.1 is executed with the sampled vertex v , an upper bound $\ell = \frac{2k}{\epsilon d_{\text{avg}}}$ an upper bound $t = k - 1$, a set F set to \emptyset , and the direction $\sigma = -$, by Lemma 4.6 the procedure returns `TRUE` under these conditions, and the graph is rejected.

4.1.3 Query complexity and running time of Algorithm 4.1

First, we prove the next lemma regarding the running time of Procedure 4.1:

Lemma 4.7. *The running time of Procedure 4.1 when given an upper bound ℓ on the number of vertices and an upper bound t on the size of the edge-cut is $O(\ell^{t+2})$.*

Proof. The recursive formula for the running time is $T(\ell, t) = \ell \cdot T(\ell, t-1) + O(\ell^2)$, since for each edge in the BFS tree (there are ℓ edges in a tree of $\ell+1$ vertices) the procedure is called with an upper bound ℓ on the number of vertices and an upper bound $t-1$ on the size of the edge-cut. The base case is $t=0$, so $T(\ell, 0) = O(\ell^2)$.

The solution is $T(\ell, t) = O(\ell^{t+2})$. We prove that $T(\ell, t) \leq c(\ell+1)^{t+2}$, where c is some constant, by induction on t . Basis: $t=0$, indeed $T(\ell, 0) = O(\ell^2) \leq c(\ell+1)^2$ for the appropriate constant c . Induction step: when $t=r$ there are ℓ recursive calls of DirectedEdgeExhaustSearch $T(\ell, r-1)$, each having a running time bounded by $c(\ell+1)^{r+1}$ according to the induction hypothesis. In addition to one BFS execution of $O(\ell^2)$ the total running time is bounded by $c\ell^2 + \ell c(\ell+1)^{r+1} \leq c(\ell+1)^{r+2}$ (the inequality is true for $r \geq 1$). \square

The number of sampled vertices is $\Theta(\frac{k}{\epsilon d_{\text{avg}}})$. For each vertex, Procedure 4.1 is run with an upper bound $\ell = \frac{2k}{\epsilon d_{\text{avg}}}$ on the number of vertices the BFS reaches and a limit $t = k-1$. According to Lemma 4.7 the running time of Procedure 4.1 is $O((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1})$ for each vertex. We get that the total complexity is $O((\frac{ck}{\epsilon d_{\text{avg}}})^{k+2})$. This can be improved to $O((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1} \log(\frac{k}{\epsilon d_{\text{avg}}}))$ using the same technique as in Algorithm 3.2 (but the improvement becomes negligible as k increases).

4.2 Testing k -edge-connectivity in bounded-degree digraphs

For the bounded-degree case we show that by using the same testing algorithm as in the unbounded-degree case, with a distance parameter that is a function of the original ϵ , we get a testing algorithm for k -edge-connectivity in bounded-degree graphs. Specifically, we run Algorithm 4.1 with a distance parameter set to $\frac{\epsilon}{13}$ and with d_{avg} set to the degree bound d .

In order to establish the above, we introduce two more notations: For a given graph G , let $a_k(G)$ denote the minimal number of edge additions that make the graph k -edge-

connected, and let $a_k^d(G)$ denote the minimal number of edge modifications that make the graph k -edge-connected, while preserving the degree bound d .

Recall that in the bounded-degree model, a graph is ϵ -far from being k -edge-connected if more than ϵdn edge modifications (additions and deletions) are necessary in order to make the graph k -edge-connected, *while preserving the degree bound*. That is, using the above notation, If graph G whose degree is bounded by d is ϵ -far from being k -edge-connected (in the bounded-degree model), then $a_k^d(G) > \epsilon dn$. Yoshida and Ito [YI09] showed that $a_k^d(G) \leq 13a_k(G)$. Consequently, if G is ϵ -far from being k -edge-connectivity in the bounded-degree model, then $a_k(G) > \frac{\epsilon}{13}dn$. This implies that we can use Algorithm 4.1 (for testing k -edge-connectivity in the unbounded-degree model) for testing k -edge-connectivity in the bounded-degree model (by running it with a distance parameter set to $\frac{\epsilon}{13}$ and with d_{avg} set to d).

The difference between the unbounded-degree case and the bounded-degree is that the number of edges between t vertices is at most $2td$ (as opposed to $t(t-1)$). The number of sampled vertices is still $\Theta(\frac{k}{\epsilon' d_{\text{avg}}})$ (where $\epsilon' = \frac{\epsilon}{13}$ and d_{avg} is d), but the BFS running time is now upper bounded by $O(\frac{k}{\epsilon'})$. Thus, the recursive formula for the running time of Procedure 4.1 with an upper bound ℓ on the number of vertices and an upper bound t on the size of the edge-cut is $T(\ell, t) = \ell \cdot T(\ell, t-1) + O(\ell d)$. The base case is $T(\ell, 0) = O(\ell d)$. The solution is $T(\ell, t) = O(\ell^{t+1}d)$. We hence get that the total complexity is $O((\frac{ck}{\epsilon d})^{k+1}d) = O((\frac{ck}{\epsilon d})^{k+1}d)$. This can be improved to $O((\frac{ck}{\epsilon d})^k d \log(\frac{k}{\epsilon d}))$ by using the same technique as in Algorithm 3.2.

4.3 Simplifying the correctness analysis of the testing algorithm for undirected k -edge-connectivity

Here we present a simplification of one of the main building blocks of the analysis of the algorithms for testing k -edge-connectivity in undirected graphs [GR02], where, for the sake of simplicity, we deal with the unbounded-degree case (addressed in [PR02]). While there are different algorithms for $k = 2$, $k = 3$ and $k \geq 4$, which employ different algorithmic techniques, they are all based on the claim that a graph that is far from

being k -edge-connected contains “many” subsets C that are “small” and such that (for $\bar{X} \triangleq V \setminus X$):

- $|E(C, \bar{C})| = r < k$;
- for every $C' \subset C$, $|E(C', \bar{C}')| > r$.

We say in this case that C is r -extreme.

The following theorem of Watanabe and Nakamura [WN87] deals with the augmentation problem of the k -edge-connectivity property (where $d(X) = |E(X, \bar{X})|$).

Theorem 4.8 ([WN87]). *An undirected graph G can be made k -edge-connected (for $k \geq 2$) by adding at most m^* new edges if and only if $\sum(k - d(X_i)) \leq 2m^*$ for every subpartition $\{X_1, \dots, X_t\}$ of V .*

By setting m^* to ϵm we get a necessary and sufficient condition for an undirected graph being ϵ -close to k -edge-connectivity. By negating the condition we get a necessary and sufficient condition for being ϵ -far from the property of k -edge-connectivity:

Corollary 4.9. *An undirected graph G is ϵ -far from being k -edge-connected if and only if there exists a partition $\{X_1, \dots, X_t\}$ of V for which $\sum(k - d(X_i)) > 2\epsilon m$.*

By applying Corollary 4.9 we can reach a similar conclusion as in Lemma 4.4:

Corollary 4.10. *In a graph that is ϵ -far from k -edge-connectivity there are at least $\frac{2\epsilon m}{k}$ disjoint subsets with an edge-cut smaller than k .*

Proof. The corollary immediately follows from the theorem: Since each subset X_i that contributes a positive value to the sum in the theorem contributes at most k , there are at least $\frac{2\epsilon m}{k}$ subsets for which $d(X) < k$. □

Using a counting argument similar to the one applied in the proof of Corollary 3.3 we get that:

Corollary 4.11. *There are at least $\frac{\epsilon m}{k}$ subsets of size at most $\frac{k}{\epsilon d_{\text{avg}}}$ with an edge-cut smaller than k .*

In addition, the next claim follows from a simple inductive argument.

Claim 4.12. *Each subset X with an edge-cut smaller than k contains a minimal subset $X' \subseteq X$, which is r -extreme for some $r < k$.*

Proof. By induction on the size of X . Base of induction: $|X| = 1$, the claim is trivially true. Induction hypothesis: the claim is true for $|X| < s$. Induction step: $|X| = s$. If all subsets of X have an edge-cut of size at least k , then X is l -extreme for some $l < k$. Otherwise, X contains a smaller subset $X' \subset X$ with an edge-cut smaller than k , so we can apply the induction hypothesis on X' . \square

We have established that a graph that is ϵ -far from k -edge-connectivity obeys the conditions that are necessary for the correctness of the testing algorithm(s). Namely, it has "many" (at least $\frac{\epsilon m}{k}$) "small" (of size at most $\frac{k}{\epsilon d_{\text{avg}}}$) subsets with an edge-cut smaller than k , for which each strict subset has an edge-cut of size at least k . This simple (given the theorem from [WN87]) proof is significantly more concise than the proof presented in [GR02].

Chapter 5

k -vertex-connectivity

The next property we study is k -vertex-connectivity in digraphs. A digraph is k -vertex-connected if for every (ordered) pair of vertices (u, v) there are k vertex-disjoint (directed) paths from u to v . Equivalently, a digraph is k -vertex-connected if for every subset X : $|\Gamma^+(X)| \geq k$ and $|\Gamma^-(X)| \geq k$.

As noted in the introduction, an algorithm for testing k -vertex-connectivity in undirected bounded-degree graphs for $k = 2, 3$ was given in [GR97]. This result was generalized to any k in [YI08] (where the dependence on k is exponential).

In this section we describe and analyze a tester for digraphs, both for the bounded-degree case and for the unbounded-degree case. In addition, we extend the result for undirected graphs and show a tester for unbounded-degree graphs as well as simplify the proof of the testing algorithm for undirected (bounded-degree) graphs.

5.1 Testing k -vertex-connectivity in digraphs

5.1.1 The testing algorithm

The idea behind the testing algorithm is similar to the one behind the testing algorithm for directed edge connectivity. We can show that a graph that is far from being k -vertex-connected has “many” subsets that are “small” and have a vertex-cut smaller than k . Our testing algorithm exploits this fact and tries to find at least one such subset, which provides evidence that the graph is not k -vertex-connected.

The main building block of the testing algorithm is Procedure 5.1 which is a variant of the algorithm “ExhaustSearch” presented in [YI08]. Similarly to the case of k -edge-connectivity, the procedure receives as input a vertex v , an upper bound, denoted ℓ , on the number of vertices that should be reached, an upper bound on the size of the vertex-cut, denoted t , the direction to work on, denoted σ (where $\sigma \in \{-, +\}$), and a forbidden subset of vertices, denoted F . The procedure is initially called with $t = k - 1$ and $F = \emptyset$. The procedure decides if a given vertex v belongs to a subset S of size at most ℓ such that $|\Gamma_{G_{V \setminus F}}^\sigma(S)| \leq t$ (where $\Gamma_{G_{V \setminus F}}^\sigma(S)$ denotes the set of vertices that are endpoints of edges with direction σ that cross the cut $E(S, \bar{S})$ in the subgraph of G induced by $V \setminus F$). The procedure returns TRUE if and only if the vertex v belongs to such a subset. Otherwise, it returns FALSE. The procedure works by running a BFS recursively. In each level of the recursion it removes a single vertex it reached, and calls itself recursively with $t - 1$ as the upper bound on the size of the vertex-cut.

Procedure 5.1: Deciding if a vertex belongs to a small set with a small vertex-cut (input: v, ℓ, t, σ, F)

If $\sigma = +$ perform the following on incoming edges, otherwise on outgoing edges:

- 1. Perform a BFS from v with the restriction that no vertex in F is passed, until $(\ell + 1)$ vertices have been reached. Let X be the set of vertices reached.*
- 2. If the BFS reached a dead-end before reaching $\ell + 1$ vertices, then return TRUE.*
- 3. If $t = 0$, then return FALSE.*
- 4. For each vertex $u \in X \setminus \{v\}$ run Procedure 5.1 with parameters $v, \ell, t - 1, \sigma$ and $F \cup \{u\}$. If any execution returns TRUE, then return TRUE. Otherwise, return FALSE.*

Algorithm 5.1 Testing k -vertex-connectivity in unbounded-degree digraphs

GENERALTESTKVERTEXCONN(distance parameter ϵ , average degree $d_{\text{avg}} = \frac{m}{n}$):

1. Sample $s = \Theta\left(\frac{k}{\epsilon d_{\text{avg}}}\right)$ vertices uniformly and independently.
 2. For each sampled vertex v run Procedure 4.1 with parameters v , $\ell = \frac{2k}{\epsilon d_{\text{avg}}}$, $t = k - 1$, $F = \emptyset$ twice: once with $\sigma = -$ and once with $\sigma = +$.
 3. If one of the executions of Procedure 4.1 returns TRUE, then REJECT.
 4. If no step caused rejection, then ACCEPT.
-

5.1.2 Correctness proof of Algorithm 5.1

At the core of our analysis is a theorem of Frank and Jordan [FJ95]. In order to state it, we present a few definitions. An ordered pair (X, Y) where $\emptyset \neq X, Y \subset V$ and $X \cap Y = \emptyset$ is a *one-way pair* in a digraph $G = (V, E)$ if there is no edge in E with a tail in X and head in Y . The *deficiency* of a one-way pair - with respect to k -vertex-connectivity - is $p_{\text{def}}(X, Y) := (k - |V \setminus (X \cup Y)|)^+$, where $(x)^+ := \max\{x, 0\}$ for some real number x . Two pairs $(X_1, Y_1), (X_2, Y_2)$ are *independent* if either $X_1 \cap X_2 = \emptyset$ or $Y_1 \cap Y_2 = \emptyset$.

Theorem 5.2 ([FJ95]). *A digraph $G = (V, E)$ can be made k -vertex-connected by adding at most m^* new edges if and only if $\sum_{(X,Y) \in \mathcal{F}} p_{\text{def}}(X, Y) \leq m^*$ holds for every family \mathcal{F} of pairwise independent one-way pairs.*

By setting m^* to ϵm we get a necessary and sufficient condition for a graph being ϵ -close to k -vertex-connectivity. By negating the condition we get a necessary and sufficient condition for being ϵ -far from the property of k -vertex-connectivity:

Corollary 5.3. *If a digraph $G = (V, E)$ is ϵ -far from being k -vertex-connected, then there exists a family \mathcal{F} of pairwise independent one-way pairs for which $\sum_{(X,Y) \in \mathcal{F}} p_{\text{def}}(X, Y) > \epsilon m$.*

The corollary implies that the (X, Y) pairs for which $p_{\text{def}}(X, Y) > 0$ are the ones in which the number of vertices that are not in $X \cup Y$ is less than k . That is, there are less than k vertices that are neither in X nor in Y (otherwise $p_{\text{def}}(X, Y) = 0$). Let's examine the pairs for which $p_{\text{def}}(X, Y) > 0$. The set of vertices that are not in $X \cup Y$ may have

an incoming edge from vertices in X or Y . No vertex in X can have an edge to a vertex in Y (according to the definition of a one-way pair), so the vertices in X can have edges to less than k vertices. In addition, the vertices in $V \setminus (X \cup Y)$ can have edges to vertices in Y and no vertex in X can have an edge to a vertex in Y (again, according to the definition of one-way pair), so the vertices in Y can have an incoming edge from at most $k - 1$ vertices. If we show that the pairs are disjoint, then we get that there are "many" subsets with a bounded-size vertex-cut.

Thus, we wish to prove that the pairs in the family of pairwise independent one-way pairs are disjoint. The next lemma refers to $\mathcal{F} = (T_1, H_1), \dots, (T_r, H_r)$, a family of pairwise independent one-way pairs of G , for which $p_{\text{def}}(\mathcal{F}) := \sum_{i=1}^r p_{\text{def}}(T_i, H_i)$ is maximized, and subject to this, $|\mathcal{F}|$ is minimized. Observe that the minimality of $|\mathcal{F}|$ implies that $p_{\text{def}}(T_i, H_i) = k - |V \setminus X \cup Y| > 0$ for each $(T_i, H_i) \in \mathcal{F}$.

Lemma 5.4 ([FJ99]). *If $p_{\text{def}}(\mathcal{F}) \geq 2k^2 - 1$, then the tails (X) are pairwise disjoint or the heads (Y) are pairwise disjoint in \mathcal{F} .*

The lemma implies that if $\epsilon m \geq 2k^2 - 1$, then these pairs are pairwise disjoint. Otherwise, $m < \frac{2k^2-1}{\epsilon}$, and then the graph is very small and can be queried completely. This gives us the next corollary:

Corollary 5.5. *A graph G that is ϵ -far from being k -vertex-connected has at least $\frac{\epsilon m}{2k}$ subsets of vertices of size at most $\frac{2k}{\epsilon d_{\text{avg}}}$, such that for each such subset S either $|\Gamma^+(S)| < k$ or $|\Gamma^-(S)| < k$.*

The corollary is proved by a simple counting argument (similar to the one in the proof of Corollary 3.3).

Similarly to what was shown in the case of k -edge-connectivity, the next claim asserts that by traversing the vertices reachable from v we can find a bounded-size vertex-cut.

Claim 5.6. *If a vertex v belongs to a subset C for which $|\Gamma^-(C)| < k$, then there exists a subset $C' \subseteq C$ such that v can reach every vertex in C' and $|\Gamma^-(C')| < k$. Analogously, if a vertex v belongs to a subset C , for which $|\Gamma^+(C)| < k$, then there exists a subset $C' \subseteq C$ such that every vertex in C' can reach v and $|\Gamma^+(C')| < k$.*

Proof. We prove for the first case of $\Gamma^-(C) < k$. A similar proof exists for the analog case of $\Gamma^+(C) < k$, where the vertices of C' can reach vertex v . We define C' all vertices in C that can be reached from v , then either $C' = C$ in which the claim holds directly by the premise on C , or there are no edge from C' to $C \setminus C'$. This means that all edges going out of C' go to $V \setminus C$, so that $\Gamma^-(C') \leq \Gamma^-(C) < k$. \square

The Correctness of Algorithm 5.1 Before proving the correctness of the testing algorithm, we establish the correctness of Procedure 5.1.

Lemma 5.7. *Suppose that Procedure 5.1 is given a vertex v that can reach, in the direction indicated by σ and without passing through any vertex in F , a set of vertices C' such that $|C'| \leq \ell$ and $|\Gamma_{G_{V \setminus F}}^\sigma(C')| \leq t$. Then the procedure returns TRUE.*

Proof. We prove the lemma by induction on $r = |\Gamma_{G_{V \setminus F}}^\sigma(C')|$. The base of induction: $r = 0$, so that the BFS surely reaches a dead-end before reaching $\ell + 1$ vertices (since $|C'| \leq \ell$), and TRUE is returned. The induction step: we prove the claim for $r > 0$, based on the induction hypothesis that the claim holds for $r - 1 \geq 0$.

The BFS runs until it reaches $\ell + 1$ vertices or it reaches a dead-end. In the latter case TRUE is returned, and it remains to deal with the former case. Let Y denote the set of vertices reached by the BFS. Since $|C'| \leq \ell$, we have that Y contains at least one vertex, denoted y , that does not belong to C' . In order to reach y , necessarily, one of the vertices $u \in \Gamma_{G_{V \setminus F}}^\sigma(C')$ had to be traversed, and thus u belongs to the set X of reached vertices. The procedure calls itself ℓ times, each time removing a different vertex from X (i.e., adding the vertex to F in the recursive call), and reducing the bound on the size of the vertex-cut. This ensures that in one of those calls, a vertex that belongs to $\Gamma_{G_{V \setminus F}}^\sigma(C')$ is removed (added to F). In this call the procedure is given v , that can reach (in the direction σ , and without traversing any vertex in F) a subset C'' of size at most $\ell - 1$, such that $|\Gamma_{G_{V \setminus F}}^\sigma(C'')| = r - 1$. For this call the induction hypothesis holds, and thus TRUE is returned. The algorithm returns TRUE if at least of the calls returned TRUE, and so TRUE is returned. \square

It is clear that the algorithm accepts any k -vertex-connected graph, since any subset of vertices has a vertex-cut of size at least k . Thus, if the graph is k -vertex-connected,

then the BFS executions (for every F they are called on, since $|F| \leq k - 1$) always reach more than $\frac{2k}{\epsilon d_{\text{avg}}}$ vertices. If the graph is ϵ -far from being k -vertex-connected, then with high constant probability, at least one vertex from a small subset X_i (for which $|\Gamma^+(X_i)| < k$ or $|\Gamma^-(X_i)| < k$) is sampled, causing Procedure 5.1 to output TRUE (based on Claim 5.6 and Lemma 5.7) and consequently the graph is rejected.

5.1.3 Query complexity and running time of Algorithm 5.1

First, we prove the next lemma regarding the running time of Procedure 5.1:

Lemma 5.8. *The running time of Procedure 5.1 given an upper bound ℓ on the number of vertices and an upper bound t on the size of the vertex-cut is $O(\ell^{t+2})$.*

Proof. The recursive formula for the running time is $T(\ell, t) = \ell \cdot T(\ell, t - 1) + O(\ell^2)$, since for each vertex reached by the BFS the procedure is called with an upper bound ℓ on the number of vertices and an upper bound $t - 1$ on the size of the vertex-cut. For the base case $t = 0$: $T(\ell, 0) = O(\ell^2)$. The solution of the formula is $T(\ell, t) = O(\ell^{t+2})$. The proof is in Lemma 4.7. \square

The number of sampled vertices is $\Theta(\frac{k}{\epsilon d_{\text{avg}}})$. For each sampled vertex, Procedure 5.1 is executed with an upper bound of $\frac{2k}{\epsilon d_{\text{avg}}}$ on the number of vertices and an upper bound $k - 1$ on the size of the vertex-cut. Setting $\ell = \frac{2k}{\epsilon d_{\text{avg}}}$ and $t = k - 1$, we get that the running time for each sampled vertex is $O((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1})$, where $c > 1$ is a constant. Therefore, the total query complexity and running time are $O((\frac{ck}{\epsilon d_{\text{avg}}})^{k+2})$. It is possible to improve the query complexity using the same technique as in Algorithm 3.2 to $O((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1} \log(\frac{k}{\epsilon d_{\text{avg}}}))$.

5.2 Testing k -vertex-connectivity in bounded-degree digraphs

In order to test k -vertex-connectivity in the bounded-degree model, we simply run Algorithm 5.1 (for unbounded-degree graphs) with distance parameter set to $\epsilon' = \frac{\epsilon}{9}$ and with d_{avg} set to d . It remains to prove that the algorithm will indeed reject with high constant probability any bounded-degree graph that is ϵ -far from being k -vertex-connectivity.

Similarly to the analysis for k -edge-connectivity in bounded-degree graphs, we denote by $a_k^d(G)$ the number of edges modification needed to make a bounded-degree graph G k -vertex-connected while preserving the degree bound, and by $a_k(G)$ the number of edges additions needed to make it k -vertex-connected while allowing any degree in the modified graph. In what follows we show that $a_k^d(G) \leq 9 \cdot a_k(G)$, which implies that we may indeed use Algorithm 5.1 for testing k -vertex-connectivity in the bounded-degree model (by executing it with the distance parameter set to $\frac{\epsilon}{9}$).

The next theorem is useful in reducing vertex degrees while preserving the vertex-connectivity.

Theorem 5.9 ([Jor93]). *Let $G = (V, E)$ be a k -vertex-connected digraph. If vertex $v \in V$ has an indegree and an outdegree of at least $k + 1$, then there exists a pair of edges (s, v) and (v, t) , such that removing those edges and adding the edge (s, t) preserves the vertex-connectivity of the graph.*

We say that a pair of edges as defined in the theorem is a *splittable pair of edges with respect to v* , and refer to the procedure of replacing the two edges with the edge (s, t) as a *split-off*. We introduce one more term: the *degree excess over d* of vertex v in a graph H is $\sum_v ((d_H^+(v) - d)^+ + (d_H^-(v) - d)^+)$ where $(x)^+ = \max(x, 0)$. We are now ready to bound the ratio $a_k^d(G)/a_k(G)$.

Lemma 5.10. *For every graph G with degree bound d we have that $a_k^d(G) \leq 9 \cdot a_k(G)$.*

Proof. Consider first adding $a_k(G)$ edges to make the graph k -vertex-connected (without necessarily maintaining the degree bound), and let the resulting graph be denoted by G' . Since it is possible that the degree of some vertices in G' is higher than d , we would like to remove all the excess over d by splitting edges incident to vertices with indegree or outdegree greater than d . In order to perform the split-offs we have to make sure that for every vertex v that violates the degree bound in G' the following inequality holds: $\min(d_{G'}^-(v), d_{G'}^+(v)) \geq k + \max(d_{G'}^-(v), d_{G'}^+(v)) - d$ (since in the splitting process we remove both one incoming edge and one outgoing edge). To this end (similarly to the proof of Lemma 3.1), we define “ports”. Each vertex has a number of ports that equals the absolute value of the difference between its indegree and outdegree in G' , namely,

$|d_{G'}^+(v) - d_{G'}^-(v)|$. The port is an *in-port* or an *out-port* depending on the direction of the deficiency (e.g., if $d_{G'}^+(v) > d_{G'}^-(v)$, then vertex v has out-ports). Now we define a complete bipartite graph $T = (A \cup B, A \times B)$, where A corresponds to the out-ports, and B corresponds to the in-ports. Note that $|A| = |B|$, since $\sum_v d_{G'}^+(v) = \sum_v d_{G'}^-(v)$, and so $\sum_{v, d_{G'}^+(v) > d_{G'}^-(v)} (d_{G'}^+(v) - d_{G'}^-(v)) = \sum_{v, d_{G'}^-(v) > d_{G'}^+(v)} (d_{G'}^-(v) - d_{G'}^+(v))$. It follows that there exists a perfect matching in T .

Since we are interested only in making sure that $\min(d_{G'}^-(v), d_{G'}^+(v)) \geq k + \max(d_{G'}^-(v), d_{G'}^+(v)) - d$ for each vertex v that has an excess over d in G' , we add only edges that correspond to the matching edges in T that cover the ports of the vertices with an (in or out) degree greater than d . We add the edges one by one. Once the inequality holds, we don't add any more edges, even if some ports are left uncovered. Let the resulting graph be denoted G'' , and let the new edges added be referred to as “the matching edges” (though the matching is in an auxiliary graph). We first claim that this addition of edges does not cause any new degree violations, that is, that the set of vertices in G'' with an indegree or an outdegree greater than d is the same as in G' . To verify this, consider any vertex v such that $d_{G'}^+(v) \leq d$ and $d_{G'}^-(v) \leq d$ and $d_{G'}^+(v) \neq d_{G'}^-(v)$ (or else v wouldn't be part of the auxiliary graph T). Assume, without loss of generality, that $d_{G'}^+(v) > d_{G'}^-(v)$ so that the number of (out-)ports that correspond to v in A is $d_{G'}^+(v) - d_{G'}^-(v)$. But then, $d_{G''}^-(v) \leq d_{G'}^-(v) + (d_{G'}^+(v) - d_{G'}^-(v)) \leq d$ (and $d_{G''}^+(v) = d_{G'}^+(v) \leq d$).

We next claim that, since $d_{G'}^+(v) \leq d$ and $d_{G'}^-(v) \leq d$, the number of matching edges added is at most $2a_k(G)$. This is true since each of the $a_k(G)$ edges added in the transformation from G to G' may increase the excess over d by at most 1 for at most two vertices, and each matching edge added (in the transformation from G' to G'') decreases the difference between the indegree and the outdegree by 1 for at least one vertex. (We note that the difference between the current argument and the one in the proof of Lemma 3.1, is that here we only add edges to vertices with a degree violation, while in the the proof of Lemma 3.1 the edges were added to all vertices with unequal indegree and outdegree.)

Now that the inequality $\min(d_{G'}^-(v), d_{G'}^+(v)) \geq k + \max(d_{G'}^-(v), d_{G'}^+(v)) - d$ holds for the vertices with degree greater than d , we can split edges until the indegree and outdegree of all vertices is at most d . The total number of edge modifications made in the split-off process is at most $6a_k(G)$. This is true because, as observed in the fore-

going discussion, for each of the $a_k(G)$ edges added in the transformation from G to G' , for at most two vertices, the excess over d is increased by 1. The addition of the matching edges (in the transformation from G' to G'') ensures the inequality holds for vertices with a degree excess. Each edge addition decreases by at least one the value of $\sum_v (k + \max(d_{G'}^-(v), d_{G'}^+(v)) - d - \min(d_{G'}^-(v), d_{G'}^+(v)))^+$. The value is at most $2a_k(G)$ in G' and 0 in G'' , so at most $2a_k(G)$ matching edges are added. The inequality holds, and the total excess is at most $4a_k(G)$ in G'' . In each split-off operation, the total excess is decreased by 2, so at most $2a_k(G)$ split-offs are performed. The number of edges modification for each split-off is 3.

Summing over all edge additions and deletions, we get that $a_k^d(G) \leq a_k(G) + 2a_k(G) + 6a_k(G) = 9a_k(G)$. \square

It remains to bound the complexity of the resulting algorithm. Since the graph G has a degree bound d , the recursive formula for the running time of Procedure 5.1 with an upper bound ℓ on the number of vertices and an upper bound t on the size of the vertex-cut is $T(\ell, t) = \ell \cdot T(\ell, t-1) + O(\ell d)$. The base case is $T(\ell, 0) = O(\ell d)$. Its solution $T(\ell, t) = O(\ell^{t+1} d)$. This gives a total query complexity and running time of $O\left(\left(\frac{ck}{\epsilon' d}\right)^{k+1} d\right)$, where $c > 1$ is a constant. Setting $\epsilon' = \frac{\epsilon}{9}$ gives $O\left(\left(\frac{ck}{\epsilon d}\right)^{k+1} d\right)$. As in previous cases, it is possible to improve the query complexity and running time to $O\left(\left(\frac{ck}{\epsilon d}\right)^k d \log\left(\frac{k}{\epsilon d}\right)\right)$ using the same technique as in Algorithm 3.2.

5.3 Reducing undirected k -vertex-connectivity to directed k -vertex-connectivity

We show a reduction from the property of k -vertex-connectivity in undirected graphs to directed graphs. This way we get a testing algorithm for undirected graphs, which is the testing algorithm we just presented for directed graphs. We deal with unbounded-degree graphs in the reduction and then add a fix for bounded-degree graphs using the split-off procedure.

The reduction is as follows: given an undirected graph $G = (V, E)$ we construct a digraph $G' = (V, E')$ with the same set of vertices. For each undirected edge $(u, v) \in E$

there are two directed edges in E' : (u, v) and (v, u) . In other words, each edge in the undirected graph G becomes a pair of anti-parallel edges in the directed graph G' .

We prove that this transformation preserves the distance to the property. First observe that if the graph G is k -vertex-connected, then clearly the digraph G' is k -vertex-connected as well, since each path becomes two paths in opposite directions. To show that if G is ϵ -far from k -vertex-connectivity then so is G' , we establish the contrapositive statement. That is, suppose that the digraph G' is ϵ -close to k -vertex-connectivity. That is, at most ϵm edges need to be added to make it k -vertex-connected. Adding the exact same edges (without a direction) to the graph G makes it k -vertex-connected as well. The reason is that each undirected edge added in G can be used in two directions, while in G' it is used in only one direction.

We conclude from this reduction that an undirected graph can be tested for k -vertex-connectivity using the testing algorithm for k -vertex-connectivity in digraphs. The algorithm treats each undirected edge in G as a pair of anti-parallel edges. The distance parameter ϵ remains the same, and hence the complexity is the same.

5.4 Simplifying the proof for undirected k -vertex-connectivity

Yoshida and Ito [YI08] presented a testing algorithm for k -vertex-connectivity in bounded-degree undirected graphs. We generalize their result for unbounded-degree graphs and simplify their proof of the algorithm.

Our proof is based on theorem of Jordan and Jackson [JJ05]. Before presenting the theorem we introduce the following notations:

1. Let $a_k(G)$ denote the minimal number of edges that have to be added to the graph to make it k -vertex-connected.
2. Let $t(G) = \max \{ \sum_{i=1}^r k - |\Gamma(X_i)| : X_1, \dots, X_r \text{ are pairwise disjoint subsets in } V \}$.
3. For $K \subset V$ let $b(K)$ denote the number of components in the subgraph induced by $V \setminus K$ and let $b(G) = \max \{ b(K) : K \subset V, |K| = k - 1 \}$.

4. Let $\delta(K) = \max \{0, \max \{k - d(x) : x \in K\}\}$.

5. Let $b^*(K) = b(K) + \delta(K)$, and $b^*(G) = \max \{b^*(K) : K \subset V, |K| = k - 1\}$.

Now that we have the notations, we can quote the theorem:

Theorem 5.11 ([JJ05]). *If G is ℓ -vertex-connected and $a_k(G) \geq 10(k - \ell + 2)^3(k + 1)^3$, then $a_k(G) = \max \{b^*(G) - 1, \lceil t(G)/2 \rceil\}$.*

Consider a graph G that is ϵ -far from k -vertex-connectivity, so that $a_k(G) \geq \epsilon m$. The theorem holds for $\epsilon m \geq 10(k - \ell + 2)^3(k + 1)^3$. Otherwise, $m < c \frac{k^6}{\epsilon}$ for some constant c , and the whole graph can be queried. Hence, from this point on we assume that $m = \Omega(\frac{k^6}{\epsilon})$.

We examine two cases:

1. $a_k(G) = \lceil t(G)/2 \rceil$. This is the simpler case, since we get that $t(G) \geq 2\epsilon m - 1$. It follows that there are at least $\frac{2\epsilon m - 1}{k}$ disjoint vertex subsets with a vertex-cut smaller than k . Using a simple counting argument (similar to the one in Corollary 3.3), we get that there are at least $\frac{\epsilon m - 1}{k}$ subsets of size at most $\frac{kn}{\epsilon m}$ with a bounded-size vertex-cut. The algorithm can sample $\Theta(\frac{k}{\epsilon d_{\text{avg}}})$ vertices uniformly and run ExhaustSearch [YI08] to find out if they belong to a small subset with a bounded-size vertex-cut.
2. $a_k(G) = b^*(G) - 1$. This means that there is a set of vertices K for which $b(K) + \delta(K) - 1 \geq \epsilon m$. $\delta(K)$ can get a maximum value of k , so the following inequality holds: $b(K) \geq \epsilon m + 1 - k$. According to the definition of $b(K)$, there are more than $\epsilon m + 1 - k$ disjoint subsets of vertices, that have a vertex-cut smaller than k . Applying a simple counting argument (similar to the one in Corollary 3.3) we get that there are at least $\frac{\epsilon m + 1 - k}{k}$ subsets of size at most $\frac{2kn}{\epsilon m}$ with a bounded-size vertex-cut. This is what we need for a testing algorithm.

The total running time of the testing algorithm for unbounded-degree graphs is $O((\frac{ck}{\epsilon d_{\text{avg}}})^{k+2})$, for some constant $c > 1$. The number of vertices sampled is $\Theta(\frac{k}{\epsilon d_{\text{avg}}})$. The running time of ExhaustSearch is $O((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1})$, when called with an upper bound $\ell = \frac{ck}{\epsilon d_{\text{avg}}}$ and a limit $t = k - 1$. The total running time can be improved to $O((\frac{ck}{\epsilon d_{\text{avg}}})^{k+1} \log(\frac{k}{\epsilon d_{\text{avg}}}))$ using the same technique as in Algorithm 3.2.

So far we showed that the conditions for testing unbounded-degree graphs hold in ϵ -far graphs: there are “many” “small” subsets with a bounded-size vertex-cut. Now, we consider the case of bounded-degree graphs. We use the same technique as before - showing that the number of edge modifications (addition and removals) necessary to make a graph k -vertex-connected while preserving the degree bound (denoted $a_k^d(G)$) is upper bounded by a multiplication of the number of edge additions without preserving the bound (denoted $a_k(G)$).

Yoshida and Ito [YI08] showed that $a_k^d(G) \leq 13 \cdot a_k(G)$, when $d \geq k + 1$ (the case of $d = k$ is treated independently in [YI08]). We immediately get a testing algorithm for bounded-degree graphs, when the degree bound is greater than k . The testing algorithm is the same algorithm as for the unbounded-degree case, but with a distance parameter $\epsilon' = \frac{\epsilon}{13}$. Its running time is $O((\frac{ck}{\epsilon d})^{k+1} d)$, which is the number of sampled vertices ($\Theta(\frac{k}{\epsilon d})$) multiplied by the running time of ExhaustSearch, which is $O((\frac{ck}{\epsilon d})^k d)$ when called with an upper bound $\ell = \frac{ck}{\epsilon d}$ and a limit $t = k - 1$. This can be improved to $O((\frac{ck}{\epsilon d})^k d \log(\frac{k}{\epsilon d}))$ using the same technique as in Algorithm 3.2.

Chapter 6

(k, ℓ) -edge-connectivity orientability

The last property we study is (k, ℓ) -edge-connectivity orientability, which we denote by (k, ℓ) -ec-orientability. A digraph is (k, ℓ) -edge-connected if there exists a vertex s , from which there are k edge-disjoint paths to any other vertex, and for any vertex $v \in V \setminus s$ there are ℓ edge-disjoint paths to s . An undirected graph $G = (V, E)$ is (k, ℓ) -ec-orientable, if there exists an orientation of the set of edges E such that the resulting digraph is (k, ℓ) -edge-connected. There is an equivalent characterization that does not involve orientation: a graph is (k, ℓ) -tree-connected if the removal of any ℓ edges leaves k edge-disjoint spanning trees in the graph. This is equivalent to (k, ℓ) -ec-orientable for $k \geq \ell$.

The case of $\ell = k$ is easily testable since we have the following theorem:

Theorem 6.1 ([NW60]). *A graph $G = (V, E)$ is (k, k) -ec-orientable if and only if G is $2k$ -edge-connected.*

Since we already know of a testing algorithm for k -edge-connectivity [GR02], we have a testing algorithm for (k, k) -ec-orientability.

For a partition $F = \{X_1, \dots, X_t\}$ of V , the corresponding *co-partition* of V is $\{V \setminus X_i, 1 \leq i \leq t\}$. For a set of subsets $T = \{Y_1, \dots, Y_t\}$ of V , a *cross edge* (with respect to T) is an edge (u, v) , such that $u \in Y_i, v \in Y_j$ and $i \neq j$. We denote the number of cross edges with respect to T by $e_G(T)$. For a partition F of V , we let $\delta(F) = k(t-1) - e_G(F)$ denote the *deficiency* of G . The *worst* partition of V is defined as the partition which has the maximum value of $\delta(F)$, that is, $\operatorname{argmax}_F \{\delta(F)\}$.

We start by citing a theorem regarding the augmentation problem of (k, ℓ) -ec-orientability:

Theorem 6.2 ([FK01]). *Let $G = (V, E)$ be an undirected graph. It is possible to add at most m^* new edges to G so that the resulting graph G^+ is (k, ℓ) -ec-orientable if and only if*

1. $m^* \geq k(t - 1) + \ell - e_G(F)$ holds for every partition F of V , where $t = |F|$.
2. $2m^* \geq t_1k + t_2\ell - e_G(F)$ hold whenever F is the union of a partition F_1 of a subset $Z \subseteq V$ and a co-partition F_2 of Z so that $|F_i| = t_i (i = 1, 2)$ and so that F_1 is a finer partition of Z than the partition $\{X : V \setminus X \in F_2\}$.

The proof methods of this theorem are constructive from an algorithmic point of view, thus they give rise to polynomial algorithms for finding a feasible augmentation.

From this point onward we focus on the case that $\ell = 0$. That is, we are interested in determining whether an undirected graph G can be oriented so that for some vertex s and every vertex v , there are k -edge-disjoint paths from s to v . We shall say in such a case that G is k -ec-orientable. By setting $m^* = 0$ (and $\ell = 0$) in Theorem 6.2 we obtain a sufficient and necessary condition for a graph to be k -ec-orientable. Note that the second condition in the theorem is equivalent to the first when $m^* = 0$ and $\ell = 0$.

Corollary 6.3. *A graph $G = (V, E)$ is k -ec-orientable if and only if for every partition F of V we have that $\delta(F) \leq 0$.*

In addition, by setting $m^* = \epsilon m$ we get a necessary and sufficient condition for a graph being ϵ -far from k -ec-orientability.

Corollary 6.4. *A graph $G = (V, E)$ is ϵ -far from k -ec-orientability if and only if there exists a partition F of V , such that $\delta(F) > \epsilon m$.*

Note that the negation of the second condition in Theorem 6.2 is stronger than the first for $\ell = 0$. So, if the negation of the second condition holds, then the negation of the first condition holds as well.

In what follows, we first observe that testing $(1, 0)$ -ec-orientability is equivalent to testing connectivity. We then give a characterization of the worst partition and finally give a linear lower bound on 1-sided-error testing of k -ec-orientability.

We show that $(1, 0)$ -ec-orientability is testable.

Theorem 6.5. *A graph $G = (V, E)$ is $(1, 0)$ -ec-orientable if and only if G is connected.*

Proof. Assume G is $(1, 0)$ -ec-orientable. Then, there is an orientation G^+ , such that in G^+ there exists a vertex s , which can reach any other vertex. So, G is connected. Assume G is connected. Examine a BFS tree from some vertex s . Orient the edges in the direction of the BFS tree, meaning if vertex u is a parent of v , then orient the edge from u to v . s can reach any vertex in this oriented graph. \square

So, testing $(1, 0)$ -ec-orientability can be done by testing if a graph is connected. A connectivity tester is given in [GR02].

6.1 A characterization of the worst partition

By Theorem 6.4 we know that graphs that are ϵ -far from k -ec-orientability have a partition F for which $\delta(F) > \epsilon m$. This inequality clearly holds for the worst partition of the graph, so we look for a characterization of the worst partition. We use this characterization in the proof of the lower bound for 1-sided-error testing algorithm and it might be useful for a 2-sided-error algorithm.

We say that a subset S of vertices is a *satisfying subset* if for every partition F of the graphs induced by S we have that $\delta(F) \leq 0$. In other words, any partition of the subset into t smaller subsets has at least $k(t - 1)$ cross edges. In addition, we define for each vertex v a *maximum subset*. The maximum subset of v is a satisfying subset, which contains v , and has the greatest number of vertices out of all satisfying subsets that contain v . We prove that a maximum subset for vertex v is unique using the following claim:

Claim 6.6. *Let S_1 be a satisfying subset and let vertex v be in S_1 . Let S_2 be a satisfying subset containing v , where $S_1 \neq S_2$. Then, $S_1 \cup S_2$ is a satisfying subset.*

Proof. Consider any partition F of the set $S_1 \cup S_2$ (with at least two subsets). Let t_1 be the number of subsets in F that have some intersection with S_1 , and let t_2 be the number of subsets that only intersect S_2 (so that $t_1 \geq 1$ while t_2 may be 0 (though $t_1 + t_2 > 1$)).

That is, F has the form $(W_1, \dots, W_{t_1}, U_1, \dots, U_{t_2})$ where: (1) each W_i is the (not necessarily disjoint) union of W_i^1 , which is a (non-empty) subset of S_1 , and W_i^2 , which is a (possibly empty) subset of S_2 , and (2) each U_j is a subset of $S_2 \setminus S_1$.

Let F_1 be the partition $(W_1^1, \dots, W_{t_1}^1)$ of S_1 . Since S_1 is a satisfying subset, $e_G(F_1) \geq k(t_1 - 1)$. Let W' be the union of all W_i^2 . Since S_1 and S_2 intersect, W' is non-empty. Now consider the partition $F_2 = (W', U_1, \dots, U_{t_2})$ of S_2 . Since S_2 is a satisfying subset, $e_G(F_2) \geq k((t_2 + 1) - 1)$.

Observe that $e_G(F)$ is the union of: (1) $e_G(F_1)$ (the edges between the subsets in F where both end-points belong to S_1); (2) $e_G(F_2)$ (the edges between the subsets in F where both end-points belong to S_2 and at least one end-point belongs to $S_2 \setminus S_1$); (3) edges between subsets such that one end-point is in $S_1 \setminus S_2$ and the other is in $S_2 \setminus S_1$; Since the three sets of edges defined above are disjoint, we have that $e_G(F) \geq e_G(F_1) + e_G(F_2) \geq k(t_1 + t_2 - 1)$, as required. \square

Corollary 6.7. *For every vertex v in V its maximum subset is unique.*

Proof. Assume, contrary to the claim, that we have two intersecting satisfying subsets, S_1 and S_2 . Both subsets contain v and have a maximum number of vertices, out of all the satisfying subsets of v .

According to Claim 6.6 $S_1 \cup S_2$ is a satisfying subset with a greater number of vertices, contradicting the fact that S_1 and S_2 have the maximum number of vertices out of all the satisfying subsets containing v . \square

Now that we have the notion of a maximum subset and we proved it is unique for each vertex, let's look at the partition in which each maximum subset is a member of the partition. Meaning, the partition is a set of all maximum subsets. We call this partition the *maximum partition*.

Theorem 6.8. *The partition of all maximum subsets is the worst partition.*

Proof. Let F be the worst partition of $G = (V, E)$. Clearly, every member of this partition is a satisfying subset, since otherwise we could find a worse partition by subdividing this subset.

Now, let $F^*(G)$ denote the maximum partition of G (that is, the partition of all maximum subsets). We prove that $\delta(F) \leq \delta(F^*(G))$ for the worst partition F of G . According to Claim 6.6, the maximum subset is unique. So, F must be a subpartition of F^* . Let's assume the contrary, meaning there exist a subset X in F , which is not contained in a subset of F^* . Let Y be a subset in F^* , which includes at least one vertex of X (there must be at least one, since both F and F^* are partitions of V). We get two satisfying subsets, X and Y , with at least one common vertex. Their union $X \cup Y$ must be a satisfying subset according to Claim 6.6, and since Y is maximal, then $(X \cup Y) \subseteq Y$, which means that X is a subset of Y contradicting the assumption.

Now, clearly a subpartition of F^* cannot be worse than F^* , since any subset of F^* is a satisfying subset. So, $\delta(F) \leq \delta(F^*)$. We get that F^* is the worst partition.

□

6.2 A lower bound for 1-sided-error testing algorithms

We prove a 1-sided-error lower bound for testing k -ec-orientability in bounded-degree graphs for $k \geq 2$. Namely, we prove that any algorithm that tests the property and accepts every graph that has the property with probability 1, must make $\Omega(n)$ queries.

Theorem 6.9. *Every 1-sided-error algorithm for testing the property of k -ec-orientability must make $\Omega(n)$ queries for $k \geq 2$.*

Proof. We start by giving the high-level idea of the proof. We define a graph that is far from being k -ec-orientable. However, if the algorithm queries βn vertices for some sufficiently small constant β , then there exists an augmentation of the subgraph that the algorithm viewed, such that the augmented graph is k -ec-orientable. Thus, any 1-sided-error algorithm must accept the graph based on its view, but this implies that it accepts graphs that are far from being k -ec-orientable. We next provide full details.

Let G_d denote a d -regular edge-expanding graph. The graph is a $(\beta, 1)$ -edge-expander for some constant β , which means that for every subset S of vertices such that $|S| \leq \beta|V|$, we have that $d(S) \geq |S|$. The existence of G_d , where $d \geq 3$, is prove by the next theorem:

Theorem 6.10 ([HLW06]). *Let $d \geq 3$ be a fixed integer. Then, for every $\delta > 0$ there exists $\alpha > 0$ such that for almost every d -regular bipartite graph G with $\frac{n}{2}$ vertices on each side, $\min_{S \subset V, |S| \leq \alpha \frac{n}{2}} \{|\Gamma(S)|\} \geq (d - 1 - \delta)|S|$.*

Corollary 6.11. *For every $d \geq 3$ there exists a d -regular graph G_d with n vertices such that for some constant β it holds that $d(S) \geq |S|$ for every $S \subset V$ which satisfies $|S| \leq \beta n$.*

Proof. According to Theorem 6.10 there exists $\beta = \frac{\alpha}{2}$ for $\delta = 1$ such that for any $|S| \leq \beta n$: $|\Gamma(S)| \geq (d - 2)|S|$. Since $d \geq 3$ and the number of neighbors is a lower bound on the number of edges we get that $d(S) \geq |S|$. \square

We first prove that graph G_{2k-1} is far from being k -ec-orientable.

Claim 6.12. *Graph G_{2k-1} is $\Omega(1/k)$ -far from k -ec-orientability.*

Proof. To prove that graph G_{2k-1} is $\Omega(1/k)$ -far from being k -ec-orientable, we show that for $\epsilon \leq \frac{0.99}{2k-1}$ and $n > 200k$, there exists a partition F for which $\delta(F) = \frac{n}{2} - k > \epsilon n$. Consider the partition F in which every vertex is a singleton subset. Namely, $F = \{\{v\} : v \in V\}$, so that $|F| = n$. The number of cross edges with respect to F is the number of edges in the graph. This equals the sum of all degrees divided by 2: $e_G(F) = n \cdot (2k - 1) \cdot \frac{1}{2}$. We get that $\delta(F) = k(n - 1) - nk + \frac{n}{2} = \frac{n}{2} - k$. This is greater than ϵn for $n > 200k$ and $\epsilon \leq \frac{0.99}{2k-1}$. \square

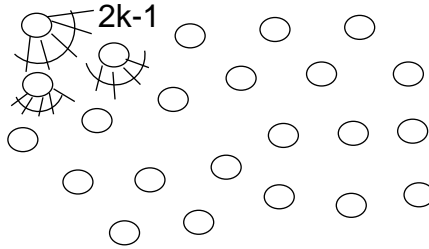


Figure 6.1: The “bad” partition of graph G_{2k-1} . All subsets are single vertices

Consider the execution of a testing algorithm A on graph G_{2k-1} , and assume that A made βn queries. The algorithm’s knowledge graph (the graph of the vertices and edges it has seen so far) is made up of at most βn vertices. We call these vertices *known*

vertices. We next show that the knowledge graph can be augmented so that the resulting graph is k -ec-orientable.

The augmentation $G^+ = (V, E^+)$ of $G_{2k-1} = (V, E)$ is defined as follows. Let V_1 be the set of queried vertices. Let V_2 be the set of unqueried vertices, then $V = V_1 \cup V_2$. $E^+ = E \cup E_2$. The subgraph $G = (V, E^+ \setminus E_2)$ equals to $G_{2k-1} = (V, E)$. The subgraph induced by the $(1 - \beta)n$ unqueried vertices $G_2 = (V_2, E_2)$ is k -ec-orientable. Next, we prove that G^+ is k -ec-orientable.

Claim 6.13. *The augmented knowledge graph $G^+ = (V, E^+)$ is k -ec-orientable.*

Proof. In order to prove that the augmented graph G^+ has the property, we have to show that for every partition $F = (X_1, \dots, X_t)$ of V : $\delta(F) \leq 0$. For this purpose, we use Theorem 6.8, which establishes that the maximum partition is the worst partition.

Let's examine the maximum partition F . First, it is obvious that all V_2 vertices belong to the same maximum subset, since G_2 is k -ec-orientable and thus V_2 is a satisfying subset. We denote this subset by S . The subset S may include queried vertices as well. Second, the queried vertices either belong to S or are singleton subsets. The reason is that any set of queried vertices is non-satisfying. This is true for the same reason that any $2k - 1$ -regular graph is not k -ec-orientable.

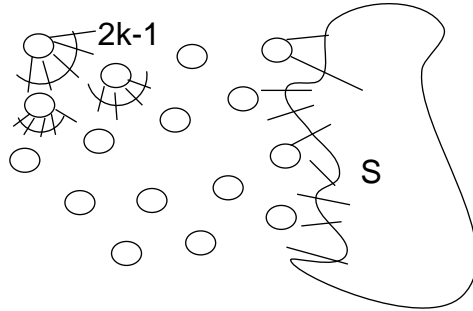


Figure 6.2: Partition F of graph G^+ . There are ℓ singleton subsets and one big subset S

Let's assume that there are ℓ singleton subsets in the maximum partition F in addition to subset S . Then, $\delta(F) = k\ell - e_G(F)$. The number of cross edges is the sum of the degrees divided by 2. Corollary 6.11 gives us that $d(S) \geq \ell$ for any $\ell \leq \beta n$, and this is true since the number of queried vertices is at most βn . Note that the lower bound applies for

$k \geq 2$, so $d = 2k - 1 \geq 3$, and thus the corollary holds. So, $e_G(F) \geq ((2k - 1)\ell + \ell)/2 = k\ell$. Following, $\delta(F) \leq k\ell - k\ell = 0$. This proves that G^+ is k -ec-orientable, since the inequality is true for the worst partition. \square

To conclude, we get that algorithm A must accept a graph, when it has queried at most βn vertices, each of degree $2k - 1$. But, this contradicts the fact that the algorithm must reject it with probability greater than $\frac{2}{3}$, since it is part of G_{2k-1} , which is far from k -ec-orientability. This proves that any 1-sided-error testing algorithm of this property must query more than βn vertices. \square

Chapter 7

Open problems

The first open question we have is: can we improve the running time of the testing algorithms for both k -vertex-connectivity and k -edge-connectivity? The running time is exponential in k , and we ask whether it is possible to make it polynomial in k . We refer to the properties in directed graphs, but this question is also relevant to k -vertex-connectivity in undirected graphs. In addition, we wonder if the property of k -vertex-connectivity is more “difficult” than k -edge-connectivity. So far, a tester whose running time is polynomial in k , was given for k -edge-connectivity in undirected graphs, but for k -vertex-connectivity the running time of the known algorithm is exponential in k .

We were also unsuccessful in coming up with a tester for the property of k -ec-orientability. Is there a 2-sided-error testing algorithm? If so, what is its running time? and is there a lower bound for a 2-sided-error algorithm?

Bibliography

- [AFKS00] N. Alon, E. Fischer, M. Krivelevich, and M Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.
- [AFNS06] N. Alon, E. Fischer, I. Newman, and A. Shapira. A combinatorial characterization of the testable graph properties: It’s all about regularity. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 251–260, 2006.
- [AS04] N. Alon and A. Shapira. Testing subgraphs in directed graphs. *Journal of Computer and System Sciences*, 69:354–482, 2004.
- [AS05a] N. Alon and A. Shapira. A characterization of easily testable induced subgraphs. *Combinatorics Probability and Computing*, 15:791–805, 2005.
- [AS05b] N. Alon and A. Shapira. A characterization of the (natural) graph properties testable with one-sided error. In *Proceedings of the Forty-Sixth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 429–438, 2005.
- [AS05c] N. Alon and A. Shapira. Every monotone graph property is testable. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on the Theory of Computing (STOC)*, pages 129–137, 2005. To appear in *SICOMP*.
- [BCL⁺06] C. Borgs, J. Chayes, L. Lovász, V. T. Sós, B. Szegedy, and K. Vesztergombi. Graph limits and parameter testing. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 261–270, 2006.
- [BR02] M. Bender and D. Ron. Testing properties of directed graphs: Acyclicity and connectivity. *Random Structures and Algorithms*, pages 184–205, 2002.

- [BSS08] I. Benjamini, O. Schramm, and A. Shapira. Every minor-closed property of sparse graphs is testable. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 393–402, New York, NY, USA, 2008. ACM.
- [CS07] A. Czumaj and C. Sohler. Testing expansion in bounded-degree graphs. In *Proceedings of the Forty-Eighth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 570–578, 2007.
- [Fis01] E. Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of the European Association for Theoretical Computer Science*, 75:97–126, 2001.
- [FJ95] A. Frank and T. Jordan. Minimal edge-coverings of pairs of sets. *Comb. Theory*, 65:73–100, 1995.
- [FJ99] A. Frank and T. Jordan. Directed vertex-connectivity augmentation. *Math. Program.*, 84:537–553, 1999.
- [FK01] András Frank and Tamás Király. Combined connectivity augmentation and orientation problems. Technical Report TR-2001-07, Egerváry Research Group, Budapest, 2001. www.cs.elte.hu/egres.
- [FN07] E. Fischer and I. Newman. Testing versus estimation of graph properties. *SIAM Journal on Computing*, 37(2):482–501, 2007.
- [Fra92] A. Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM J. on Discrete Mathematics*, 1:22–53, 1992.
- [GGR98] 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.
- [Gol98] O. Goldreich. Combinatorial property testing - a survey. In *Randomization Methods in Algorithm Design*, pages 45–60, 1998.

- [GR97] O. Goldreich and D. Ron. Property testing in bounded degree graphs. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 406–415, 1997.
- [GR99] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.
- [GR02] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, pages 302–343, 2002.
- [HKNO09] A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Proceedings of the Forty-Eighth Annual Symposium on Foundations of Computer Science (FOCS)*, 2009.
- [HLNT05] S. Halevy, O. Lachish, I. Newman, and D. Tsur. Testing orientation properties. Technical Report TR05-153, Electronic Colloquium on Computational Complexity (ECCC), 2005.
- [HLW06] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. 43(4):469–561, 2006.
- [Izb04] A. Izbinsky. Testing the diameter of directed graphs. Master’s thesis, School of Electrical Engineering, 2004.
- [JJ05] B. Jackson and T. Jordán. Independence free graphs and vertex connectivity augmentation. *J. Comb. Theory Ser. B*, 94(1):31–77, 2005.
- [Jor93] T. Jordán. Increasing the vertex-connectivity in directed graphs. In *ESA '93: Proceedings of the First Annual European Symposium on Algorithms*, pages 236–247, London, UK, 1993. Springer-Verlag.
- [KKR04] T. Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004.

- [KS07] S. Kale and C. Seshadhri. Testing expansion in bounded degree graphs. Technical Report TR07-076, Electronic Colloquium on Computational Complexity (ECCC), 2007. Available from <http://www.eccc.uni-trier.de/eccc/>.
- [NS07] A. Nachmias and A. Shapira. Testing the expansion of a graph. Technical Report TR07-118, Electronic Colloquium on Computational Complexity (ECCC), 2007. Available from <http://www.eccc.uni-trier.de/eccc/>.
- [NW60] C.S.J.A Nash-Williams. On orientations, connectivity and odd vertex pairings in finite graphs. *Canad. J. Math.*, 12:555–567, 1960.
- [PR02] M. Parnas and D. Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, 20(2):165–183, 2002.
- [Ron01] D. Ron. Property testing. In *Handbook on Randomization, Volume II*, pages 597–649, 2001. Editors: S. Rajasekaran, P. M. Pardalos, J. H. Reif and J. D. P. Rolim.
- [Ron08] D. Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008.
- [Ron10] D. Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Machine Learning*, 2010. To appear.
- [RS96] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.
- [WN87] T. Watanabe and A. Nakamura. Edge-connectivity augmentation problems. *J. Comput. Syst. Sci.*, 35(1):96–144, 1987.
- [YI08] Y. Yoshida and H. Ito. Property testing on k -vertex-connectivity of graphs. In *ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I*, pages 539–550, Berlin, Heidelberg, 2008. Springer-Verlag.
- [YI09] Y. Yoshida and H. Ito. Testing k -edge-connectivity of digraphs. To appear in *Journal of System Science and Complexity*, 2009.