Chapter 2
Property Testing on Graphs and Games

Hiro Ito

Abstract Constant-time algorithms are powerful tools, since they run by reading only a constant-sized part of each input. Property testing is the most popular research framework for constant-time algorithms. In property testing, an algorithm determines whether a given instance satisfies some predetermined property or is far from satisfying the property with high probability by reading a constant-sized part of the input. A property is said to be testable if there is a constant-time testing algorithm for the property. This chapter covers property testing on graphs and games. The fields of graph algorithms and property testing are two of the main streams of research on discrete algorithms and computational complexity. In the section on graphs in this chapter, we present some important results, particularly on the characterization of testable graph properties. At the end of the section, we show results that we published in 2020 on a complete characterization (necessary and sufficient condition) of testable monotone or hereditary properties in the bounded-degree digraphs. In the section on games, we present results that we published in 2019 showing that the generalized chess, Shogi (Japanese chess), and Xiangqi (Chinese chess) are all testable. We believe that this is the first results for testable EXPTIME-complete problems.

2.1 Introduction

The development of efficient algorithms for problems on big data problems is an urgent task. Constant-time algorithms are a powerful tool for this since they run by reading only a constant-sized part of each input. In other words, the running time is invariant regardless of the size of the input. Property testing is the most popular research framework for constant-time algorithms. In property testing, an algorithm determines whether a given instance satisfies some predetermined property or is far from satisfying that property with high probability by reading a constant-sized part of the input. This section presents some results mainly concerning property testing that have recently been obtained in the ABD Project.

H. Ito (✉)
The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan
e-mail: itohiro@uec.ac.jp

N. Katoh et al. (eds.), *Sublinear Computation Paradigm*,
https://doi.org/10.1007/978-981-16-4095-7_2

2.2 Basic Terms and Definitions for Property Testing

This section gives some of the basic terms that are needed in order to explain our results. Property testing works on many different types of models, including graphs, functions, strings, grammars, and images. Although the details of the definitions differ slightly between the different models, since the basic ideas are the same for all of models, we present only the definitions for digraphs.

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers. In this chapter, we sometimes omit floor or ceiling functions. For example, if we write $s = \sqrt{n}$ in a context where s must be an integer and n is not necessarily a square number, then $\sqrt{n}$ should be taken to mean $\lfloor\sqrt{n}\rfloor$ or $\lceil\sqrt{n}\rceil$. This allows us to disregard integrality issues that make no real difference to any of our proofs.

2.2.1 Graphs and the Three Models for Property Testing

A directed graph or *digraph* G is defined as a pair of finite sets (V, E), where V is a finite set of *vertices* and $E \subseteq V \times V$ is a set of directed edges, or *edges* for short. The vertex set V and the edge set E of a graph G are sometimes written as V_G and E_G, respectively. If the direction of each edge is ignored (i.e., $(u, v) = (v, u)$ for any $u, v \in V$), then the digraph is called a *graph* (or an *undirected graph* if we want to indicate undirectedness explicitly). Every graph can be represented as a digraph by using reflectivity on edges; in other words if $(u, v) \in E$, then $(v, u) \in E$ for every $u, v \in V$. Thus, graphs can be regarded as special cases of digraphs. This section mainly treats (undirected) graphs. Digraphs are considered in Sect. 2.4. Many of the terms and symbols we define for graphs are also used for digraphs.

The *order* of a graph G is given by $|V_G|$ and the *size* of a graph G is given by $|E_G|$. A graph (resp., digraph) of order n is also called an *n-graph* (resp., *n-digraph*). The number of vertices adjacent to a vertex v in a graph G is denoted by $\deg_G(v)$. If G is clear from the context, the subscript G may be omitted. In property testing, since an algorithm reads only a part of an instance (input), it gets information about the instances through oracles, which depend on how to the graphs are represented. There are three known models for treating graphs in property testing: the dense-graph model; the bounded-degree (graph) model; and the general-graph model.

In *the dense-graph model*, the *edge oracle* is used: If an algorithm queries whether $(u, v) \in E$ or not, the oracle answers correctly: the answer is 1 if $(u, v) \in E$ and 0 otherwise. This model basically treats dense (i.e., $|E| = \Omega(n^2)$) graphs. This is because if $|E| = o(n^2)$, then the edge oracle answers "0" almost every time when n is large, making the queries useless.[1]

In the *bounded-degree model*, there is a restriction such that the degree of every vertex is bounded by a predetermined integer $d \geq 1$, that is, $\deg(v) \leq d$ $(\forall v \in V)$. From this restriction, it follows that the number of edges in a graph is at most $dn/2$ (or

[1] It works only for determining whether a given graph is sparse.

dn for a digraph); in other words, the graph is sparse (note that d is a constant). This model assumes that for every vertex v, the vertices adjacent to v are ordered. This model uses the *adjacent-vertex oracle*: If an algorithm queries for the ith ($1 \leq i \leq d$) adjacent vertex of v by giving a pair (v, i), the oracle answers the name (ID) of the vertex if exists and returns a predetermined special symbol such as $\perp$ otherwise. A graph where the degree is bounded by d is also called a *d-bounded-degree graph*.

The *general-graph model* is a mixed model of the dense-graph model and the bounded-degree model. Although this model does not have any maximum degree-bound, there is a fixed upper bound d on the *average* degree. In many cases d is a constant and the graphs in this model are sparse. However, if $d = \Theta(n)$, graphs in the model may be dense. This model allows all oracles that are allowed in the other two models in addition to the *degree oracle*: If an algorithm queries the degree of a vertex v, it replies with the correct answer $\deg(v)$.

2.2.2 Properties, Distances, and Testers

The set of graphs considered in each model—that is, the dense-graph model, the bounded-degree model, or the general-graph model—is denoted by Γ. The subset of Γ such that the order of the graph is n is denoted by Γ_n. Hence $\Gamma = \bigcup_{n\in\mathbb{N}} \Gamma_n$.

A *property* is defined as a (generally infinitely large) subset of graphs closed under isomorphism.[2] For example "planarity" is defined as the set of all planar graphs. For a property $\mathcal{P}$, we define $\mathcal{P}_n$ as $\mathcal{P} \cap \Gamma_n$. Thus, clearly $\mathcal{P} = \bigcup_{n\in\mathbb{N}} \mathcal{P}_n$.

Property testing is a relaxation of a decision problem. The object of a property testing is to distinguish with high probability whether a given instance satisfies some predetermined property or the instance is "far" from satisfying the property. This requires a mathematical definition of "far."

Let G and G' both be n-graphs; $G, G' \in \Gamma_n$. The distance between the two graphs is defined as the Hamming distance between them divided by the largest Hamming distance in the model (for normalization). Thus, the distance depends on the models (i.e., how the graphs are represented). We explain this by using the dense-graph model. Let $\delta_{E_G} : V \times V \to \{0, 1\}$ be the characteristic function on E_G, that is, $\delta_{E_G}(u, v) = 1$ if $(u, v) \in E_G$ and 0 otherwise. The distance between G and G' is defined as follows: We denote by $m(G, G')$ the number of edges that need to be deleted from and/or inserted into G in order to make $G = G'$, i.e.

$$m(G, G') := |\{(u, v) \in V \times V \mid \delta_{E_G}(u, v) \neq \delta_{E_{G'}}(u, v)\}|$$

Using this, we define the distance between G and G' as follows[3]:

---

[2] Intuitively this means to ignore the labels on vertices and edges.

[3] Although the maximum number of edges in any (undirected) graph of order n is $n(n-1)/2$, we use n^2 for the denominator for simplicity.

$$\mathrm{dist}(G, G') := \frac{m(G, G')}{n^2}. \tag{2.1}$$

Note that $0 \leq \mathrm{dist}(G, G') \leq 1$ for every G and G'. In the bounded-degree model and the general-graph model, the distance is defined as follows[4]:

$$\mathrm{dist}(G, G') := \frac{m(G, G')}{dn}, \tag{2.2}$$

where d is the upper bound on the maximum (resp., the average) vertex-degrees for the bounded-degree model (resp., the general-graph model).

By using the distance between graphs, the distance beetween a graph $G \in \Gamma_n$ and a property $\mathcal{P}$ is defined as follows:

$$\mathrm{dist}(G, \mathcal{P}) := \begin{cases} \min_{G' \in \mathcal{P}_n} \mathrm{dist}(G, G') & \text{if } \mathcal{P}_n \neq \emptyset, \\ \infty & \text{otherwise.} \end{cases}$$

This applies to all the models. For a real value $0 \leq \epsilon \leq 1$, we say that G is *ϵ-far* from G' (resp., $\mathcal{P}$) if $\mathrm{dist}(G, G') > \epsilon$ (resp., $\mathrm{dist}(G, \mathcal{P}) > \epsilon$) and *$\epsilon$-close* otherwise.

A *testing algorithm* for a property $\mathcal{P}$ is an algorithm that, given query access (by the oracles) to an instance G and given $0 < \epsilon \leq 1$, accepts every $G \in \mathcal{P}$ with probability at least $2/3$, and rejects every G that is ϵ-far from $\mathcal{P}$ with probability at least $2/3$. If a testing algorithm accepts every $G \in \mathcal{P}$ with probability 1, then the algorithm is called a *one-sided-error*. The number of queries made by an algorithm to the given oracle is called the *query complexity* of the algorithm. If the query complexity of a testing algorithm is bounded by a constant that is independent of n (but that may depend on ϵ and d), then the algorithm is called a *tester*. A property is *testable*[5] if there is a tester for the property.

2.3 Important Known Results in Property Testing on Graphs

This section gives a very brief overview of important known results in property testing on graphs, particularly on the characterization and general properties of testability. See a recent review [11] or books [4, 8] for details.

[4] Although the maximum number of edges of any d-bounded-degree (undirected) graph of order n is $dn/2$, we use dn for the denominator for simplicity.

[5] Sometimes "testable" means that the problem has an algorithm with sublinear query complexity, and *strongly testable* may be used to distinguish constant query complexity from mere sublinear query complexity.

2.3.1 Results for the Dense-Graph Model

Alon et al. [2] found a combinatorial characterization (necessary and sufficient condition) of testable properties for the dense-graph model. We first present the theorem without defining the terms used in it.

Theorem 2.1 *For the dense-graph model, a graph property is testable if and only if it is regular-reducible.*

This theorem utilize the extremely powerful monumental Szeméredi's regularity lemma, which we now introduce briefly. For a pair of subsets of vertices $A, B \subseteq V$ of a graph $G = (V, E)$, $\mathrm{den}(A, B) := \frac{|E(A,B)|}{|A||B|}$ is called the *density* of the pair. A family of subsets $\mathcal{V} = \{V_1, \ldots, V_k\}$ ($V_i \subseteq V$, $\forall i \in \{1, \ldots, k\}$) is called a *partition* of V if $V_i \cap V_j = \emptyset$ for all $1 \leq i < j \leq k$ and $V = V_1 \cup \cdots \cup V_k$. A partition $\mathcal{V} = \{V_1, \ldots, V_k\}$ of the vertex set of a graph is called an *equipartition* if $|V_i|$ and $|V_j|$ differ by no more than 1 for all $1 \leq i < j \leq k$.

Definition 2.1 (ϵ-*regular*) Let $0 < \epsilon \leq 1$ be a real number and $A, B \subseteq V$. A pair (A, B) is called ϵ-*regular* if $|\mathrm{den}(A, B) - \mathrm{den}(X, Y)| \leq \epsilon$ for any two subsets $X \subseteq A$ and $Y \subseteq B$ satisfying $|X| \geq \epsilon|A|$ and $|Y| \geq \epsilon|B|$. An equipartition $\mathcal{V} = \{V_1, \ldots, V_k\}$ of the vertex set of a graph is called ϵ-*regular* if all but at most ϵk^2 of the pairs (V_i, V_j) ($i, j \in \{1, \ldots, k\}$) are ϵ-regular.

Definition 2.2 (*regularity-instance*) A *regularity-instance* R is given by an error-parameter $0 < \epsilon \leq 1$, an integer k, a set of $\binom{k}{2}$ real numbers $0 \leq \eta_{i,j} \leq 1$ indexed by $1 \leq i < j \leq k$, and a set $\overline{R}$ of pairs (i, j) of size at most ϵk^2. A graph is said to satisfy the *regularity-instance* if it has an equipartition $\mathcal{V} = \{V_1, \ldots, V_k\}$ such that for all $(i, j) \notin \overline{R}$ the pair (V_i, V_j) is ϵ-regular and satisfies $|E(V_i, V_j)| = \eta_{i,j}|V_i||V_j|$. The *complexity* of the regularity instance is $\max(k, 1/\epsilon)$.

Definition 2.3 (*regular-reducible*) A graph property $\mathcal{P}$ is *regular-reducible* if for any $\delta > 0$ there exists $r = r_{\mathcal{P}}(\delta)$ such that for any n there is a family $\mathcal{R}$ of at most r regularity-instances each of complexity at most r, such that the following holds for every $\epsilon > 0$ and every n-graph G:

1. If $G \in \mathcal{P}$, then for some $R \in \mathcal{R}$, G is δ-close to R.
2. If G is ϵ-far from $\mathcal{P}$, then for any $R \in \mathcal{R}$, G is $(\epsilon - \delta)$-far from R.

Theorem 2.2 (Szeméredi's regularity lemma [2, 17]) *For every pair of an integer t and a real number $\epsilon > 0$ there exists an integer $T = T_2(t, \epsilon)$ such that any graph with $n \geq T$ vertices has an ϵ-regular equipartition of order k, where $t \leq k \leq T$.*

An intuitive explanation of the regularity lemma is that, for any $\epsilon > 0$, every graph $G = (V, E)$ has an ϵ-approximation of a constant-sized edge-weighted graph, where the edge weight approximates the density of the corresponding vertex pair. Intuitively, a property being regular-reducible means that it can be represented by a

constant number of equipartitions based on the regularity lemma; in other words, the regularity lemma holds for testing the property. See [11] also for details.

Representative regular-reducible properties are monotone or hereditary properties, which are defined as follows.

Definition 2.4 A graph property $\mathcal{P}$ is *monotone* if for every $G \in \mathcal{P}$ and $e \in E_G$, $G - \{e\} \in \mathcal{P}$. A graph property $\mathcal{P}$ is *hereditary* if for every $G \in \mathcal{P}$ and $v \in V_G$, $G - \{v\} \in \mathcal{P}$.

Planarity, bipartiteness, k-colorability (for any $k \in \mathbb{N}$), H-freeness (for any graph H),[6] and disconnectedness are all monotone. The former four properties are also hereditary, but the last one, disconnectedness, is not.[7] A well-known non-monotone and hereditary property is perfectness: A graph is said to be *perfect* if for every induced subgraph, the chromatic number of the subgraph equals the clique number (= the order of the largest clique) of the subgraph. Every monotone or hereditary property is regular-reducible (see [2] for details).

We can say that Theorem 2.1 solves the problem of characterizing testable properties in the dense-graph model in a sense. However, the constants that appear in the algorithms obtained by Theorem 2.1 are incredibly (maybe more than astronomically) huge! Thus, developing faster (i.e., smaller constant complexity) algorithms remains an issue for each problem.

2.3.2 Results for the Bounded-Degree Model

Whereas the combinatorial characterization of testable properties as shown in Theorem 2.1 was obtained for the dense-graph model, no perfect results have been obtained for the bounded-degree model despite many attempts to achieve this goal. However, progress is being made in steps. We now have an important characterization of testable properties in the bounded-degree model called "hyperfiniteness." We also found another characterization called "forbidden configurations," for one-sided error testability, which is explained in Sect. 2.4.

Definition 2.5 Let $\epsilon > 0, t > 0$, and $d > 0$. Let $G = (V, E)$ be a d-bounded-degree n-graph. If one can remove at most ϵdn edges from G such that each connected component of the resulting graph has at most t vertices, then G is called (ϵ, t)-*hyperfinite* (with respect to degree bound d). For a function $\rho : \mathbb{R}^+ \to \mathbb{R}^+$, if G is $(\epsilon, \rho(\epsilon))$-hyperfinite for every $\epsilon > 0$, then G is called ρ-*hyperfinite*. A set $\mathcal{G}$ of d-degree-bounded graphs is called ρ-*hyperfinite* if $\forall G \in \mathcal{G}$ is ρ-hyperfinite. $\mathcal{G}$ is called *hyperfinite* if there is a function ρ such that $\mathcal{G}$ is ρ-hyperfinite.

Newman and Sohler [15] presented the following theorem.

---

[6] If a graph includes no H as a subgraph, then it is called H-*free*.

[7] If a graph consisting of one connected component of order $n - 1$ and one isolated vertex is disconnected, removing the isolated vertex from the graph makes it connected.

Theorem 2.3 *In the bounded-degree model, every graph property is testable for any hyperfinite family of graphs.*

While this is a sufficient condition, the following necessary condition related to hyperfiniteness was obtained by Fichtenberger et al. [5].

Definition 2.6 A *subproperty* of a property $\mathcal{P}$ is a property that is a subset of $\mathcal{P}$. A property is *non-trivially testable* if it is testable and there exists $\epsilon > 0$ such that there is an infinite number of graphs that are ϵ-far from the property.

Theorem 2.4 *Every testable property of bounded-degree graphs is either finite or contains an infinite hyperfinite subproperty. Furthermore, the complement of every non-trivially testable graph property contains an infinite hyperfinite subproperty.*

These theorems show that there is a deep relation between hyperfiniteness and testability on bounded-degree graphs. We have found, however, no necessary and sufficient condition of graph testability even for a one-sided error. Recently we found necessary and sufficient conditions for one-sided-error testability on subclasses of properties of digraphs[8] [12]. This was obtained through the ABD Project, and is explained in Sect. 2.4.

2.3.3 Results for the General-Graph Model

There were previously no general classes of testable properties for the general-graph model. Through the ABD Project, a class that models complex networks called Hierarchical Scale Free ($\mathcal{HSF}$) was founded that is testable. We present an outline of the result below, and the details are available in [10, 11].

Definition 2.7 *For positive real numbers $c > 0$ and $\gamma > 1$, a class of scale-free (multi)graphs $\mathcal{SF}(c, \gamma)$ consists of (multi)graphs $G = (V, E)$ for which the following condition holds: Let v_i be the number of vertices v of degree i. Then:*

$$v_i \leq cni^{-\gamma}, \quad \forall i \in \{2, 3, \ldots, \}. \tag{2.3}$$

A *clique* is a subgraph in which there exists an edge between every pair of vertices. For a nonnegative integer $c \geq 0$, a *c-isolated clique* is a clique such that the number of outgoing edges (edges between the clique and the other vertices) is less than ck, where k is the number of vertices of the clique. A 1-isolated clique is sometimes simply called an *isolated clique* (see [9] for details). Let $\mathcal{E}(G)$ be the graph obtained from G by contracting all isolated cliques.[9]

[8] Note that any undirected graph can be represented by a digraph, i.e., the set of digraphs can be regarded as including the set of undirected graphs.

[9] Two distinct isolated cliques never overlap, except in the special case of *double-isolated-cliques*, which consists of two isolated cliques of size k that share $k - 1$ vertices. A double-isolated-clique

Definition 2.8 For positive real numbers $c > 0, \gamma > 1$ and a positive integer $n_0 \geq 1$, a class of *hierarchical scale-free (multi)graphs* $\mathcal{HSF} = \mathcal{HSF}(c, \gamma, n_0)$ consists of (multi)graphs $G = (V, E)$ for which the following conditions hold:

(i) $G \in \mathcal{SF}(c, \gamma)$,
(ii) Consider the infinite sequence of graphs $G_0 = G, G_1 = \mathcal{E}(G_0), G_2 = \mathcal{E}(G_1), \ldots$. If $|V_{G_i}| \geq n_0$, then G_i includes at least one isolated clique $Q \subseteq V$ with $|Q| \geq 2$. (Note that if G_k has no such isolated clique, then $G_k = G_{k+1} = G_{k+2} = \cdots$.)

For a graph G and a nonnegative integer $d \geq 0$, $G|d$ is the graph obtained by deleting all edges incident to each vertex v of degree more than d. Note that $G|d$ is a d-bounded-degree graph. The following properties were obtained by [10].

Lemma 2.1 *For every $\mathcal{SF} = \mathcal{SF}(c, \gamma)$ with $\gamma > 2$, and every positive real number $\epsilon > 0$, there exists a constant $\delta = \delta(\epsilon, c, \gamma)$ such that for every graph $G \in \mathcal{SF}$, $G|\delta$ is ϵ-close to G.*

This lemma looks useful since it means that for any $\epsilon > 0$, any scale-free graph is ϵ-close to a bounded-degree graph. This lemma is applied in the proof of the following theorem, which is the main theorem of [10].

Theorem 2.5 *Every property is testable for $\mathcal{HSF}(c, \gamma, n_0)$ with $\gamma > 2$.*

In the general-graph model, no other universal (constant-time) tester is known, but universal testing algorithms with polylog(n)-time query complexity have been found for forests [14] and outerplanar graphs [3].

2.4 Characterization of Testability on Bounded-Degree Digraphs

2.4.1 Bounded-Degree Model of Digraphs

As mentioned previously, there is no complete characterization of testable graph properties in bounded-degree graphs even for one-sided-errors. Through the ABD project, however, we have obtained a characterization for one-sided-error testable properties of monotone and hereditary properties of bounded-degree digraphs [12], which we briefly explain in this section. The set of digraphs can be regarded to include the set of undirected graphs by introducing reflexivity, i.e., $\forall u, v \in V$, if $(u, v) \in E$, then $(v, u) \in E$.

In this section, we consider the bounded-degree model on digraphs. For a digraph $G = (V, E)$ and a vertex $v \in V$ we denote by $N_G^+(v)$ the set of outgoing neighbours

---

Q has no edge between Q and the other part of the graph (i.e., $\deg_G(Q) = 0$), and thus we specially define that a double-isolated-clique in G is contracted into a vertex in $\mathcal{E}(G)$. Under this assumption, $\mathcal{E}(G)$ is uniquely defined.

of v, i.e., $N_G^+(v) := \{u \in V \mid (v,u) \in E\}$. Similarly, $N_G^-(v) := \{u \in V \mid (u,v) \in E\}$ and $N_G(v) := N_G^+(v) \cup N_G^-(v)$. The *out-degree* of v is $\deg_G^+(v) := |N_G^+(v)|$, and the *in-degree* of v is $\deg_G^-(v) := |N_G^-(v)|$. The subscript G can be omitted if it is clear.

For a (di)graph $G = (V, E)$ and $F \subseteq E$, we denote by $G - F$ the graph $(V, E - F)$. For a (di)graph $G = (V, E)$ and $W \subseteq V$, we denote by $G[W]$ the subgraph of G induced by W (i.e., $G[W]$ contains all edges in E_G whose both endpoints are in W). $G[V - W]$ can be denoted by $G - W$.

In the bounded-degree model for digraphs, there are two submodels: In one, only the out-degree is bounded; in the other, both the in-degree and out-degree are bounded.[10] The former case is represented by $F(d)$ model and the latter one by $FB(d)$ model.[11] The $F(d)$ model is clealy wider than the $FB(d)$ model. Moreover, every undirected d-bounded graph can be formulated by the $FB(d)$ model by replacing each undirected edge by a pair of anti-parallel directed edges. That is, the $FB(d)$ model (and thus the $F(d)$ model as well) is regarded as including the undirected d-bounded degree model.

2.4.2 Monotone Properties and Hereditary Properties

This section extends the monotone and hereditary properties that were defined in Definition 2.4 to digraphs.

We first introduce the following notation for characterizing the testability of these properties. Let $\mathcal{H}$ be a set of digraphs. We call $\mathcal{H}$ an *r-set* if every member $H \in \mathcal{H}$ has at most r vertices (i.e., H is an r'-digraph for some $r' \leq r$). A digraph G is *$\mathcal{H}$-free* if for every $H \in \mathcal{H}$, G contains no subgraph that is isomorphic to H. A digraph G is *induced $\mathcal{H}$-free* if for every $H \in \mathcal{H}$, G contains no induced subgraph that is isomorphic to H. We denote by $\mathcal{P}_{\mathcal{H}}$ (resp., $\mathcal{P}^*_{\mathcal{H}}$) the property that contains all digraphs that are $\mathcal{H}$-free (resp., induced $\mathcal{H}$-free). $\mathcal{P}_{\mathcal{H},n}$ (resp., $\mathcal{P}^*_{\mathcal{H},n}$) is the subproperty of $\mathcal{P}_{\mathcal{H}}$ that consists of all n-digraphs in $\mathcal{P}_{\mathcal{H}}$ (resp., $\mathcal{P}^*_{\mathcal{H}}$). We can easily confirm that $\mathcal{P}_{\mathcal{H}}$ is monotone and $\mathcal{P}^*_{\mathcal{H}}$ is hereditary for any $\mathcal{H}$.

Let $H = (V, E)$ be a digraph. For a subset $W \subseteq V$, if by disregarding the directions of the edges of H, W induces a connected component in the resulting undirected graph, then we say that $H[W]$, which is the directed subgraph of H induced by W, is a *component* of H. A digraph H is *rooted* if every component H' of H has a vertex v such that for every $u \in V_{H'}$ there exists a dipath (= directed path) from v to u.

[10] Clearly the case in which only the *in*-degree is bounded can be formulated by the model in which only the *out*-degree is bounded by changing the edge direction.

[11] F and B mean forward and backward, respectively.

2.4.3 Characterizations

By using these terms, the characterizations of testable monotone or hereditary properties for the $F(d)$ model were given in [12].

Theorem 2.6 *Let $P = \bigcup_{n\in\mathbb{N}} \mathcal{P}_n$ be a monotone property in the $F(d)$-model. Then $\mathcal{P}$ is testable if and only if there is a function $r : (0, 1) \to \mathbb{N}$ such that for any $0 < \epsilon < 1$ and $n \in \mathbb{N}$, there is an $r(\epsilon)$-set of rooted digraphs $\mathcal{H}_n$ such that the property $\mathcal{P}_{\mathcal{H}_n,n}$ satisfies the following two conditions:*
(a) $\mathcal{P}_n \subseteq \mathcal{P}_{\mathcal{H}_n,n}$
(b) $\mathcal{P}_{\mathcal{H}_n,n}$ is $\epsilon/2$-close to $\mathcal{P}_n$.

Theorem 2.7 *Let $\mathcal{P}$ be a hereditary property in the $F(d)$-model. Then $\mathcal{P}$ is testable if and only if there are functions $r : (0, 1) \to \mathbb{N}$ and $N : (0, 1) \to \mathbb{N}$ such that for any $0 < \epsilon < 1$, there is an $r(\epsilon)$-set of rooted digraphs $\mathcal{H}$ such that for every $n \geq N(\epsilon)$, $\mathcal{P}^*_{\mathcal{H},n}$ satisfies the following two conditions:*
(a) $\mathcal{P}_n \subseteq \mathcal{P}^*_{\mathcal{H},n}$
(b) $\mathcal{P}^*_{\mathcal{H},n}$ is $\epsilon/2$-close to $\mathcal{P}_n$.

Condition (b) in both Theorems 2.6 and 2.7 is necessary, since there exists a monotone and hereditary property that is testable with a one-sided-error and has no $\mathcal{H}_n$ such that $|\mathcal{H}_n|$ is bounded by a constant $(r(\epsilon))$ and "$\mathcal{P}_n = \mathcal{P}_{\mathcal{H}_n,n}$ or $\mathcal{P}_n = \mathcal{P}^*_{\mathcal{H}_n,n}$": One of these properties is $\mathcal{P}_{C_{\sqrt{n}}}(= \mathcal{P}^*_{C_{\sqrt{n}}})$ on the $F(1)$-model,[12] where C_k is the set of directed cycles (or *dicycles*, for short) of length in $[3, k]$, i.e., $\mathcal{P}_{C_{\sqrt{n}}}$ is the property of having no dicycle of length in $[3, \sqrt{n}]$. This property is clearly monotone and hereditary. To express $\mathcal{P}_{C_{\sqrt{n}}}$ by using a set $\mathcal{H}$ of forbidden subgraphs (or forbidden induced subgraphs), $\mathcal{H}$ must includes $C_{\sqrt{n}}$, and thus $|\mathcal{H}|$ cannot be bounded by any constant. However, this property is testable with a one-sided-error as shown below.

Lemma 2.2 *$\mathcal{P}_{C_{\sqrt{n}}}$ on the $F(1)$-model is one-sided-error testable with query complexity $O(\epsilon^{-2})$.*

To prove this lemma, we will use the following lemma, which is often effective for estimating the query complexity of testers.

Lemma 2.3 *For any real number x, the following inequality holds:*

$$e^x \geq x + 1. \tag{2.4}$$

The proof of this lemma is trivial from the differentiation of $e^x - (x + 1)$, and is omitted here.

Proof of Lemma 2.2: If $n \leq 2/\epsilon$, then we can get the complete data of the graph in time $2/\epsilon$. Thus it is enough to consider the case of $n > 2/\epsilon$. We use the following algorithm for the tester:

[12] $\mathcal{P}_{C_{\sqrt{n}}} \neq \mathcal{P}^*_{C_{\sqrt{n}}}$ on the $F(d)$-model for $d \geq 2$.

Choose $s = 2/\epsilon$ vertices $v_1, \ldots, v_s$ from V uniformly at random, and denote them by S. For each $v_i \in S$, check whether there is a dicycle of length at most s that includes v_i by following each outgoing edge successively whenever it exists. (Note that in the $F(1)$-model, the outgoing edge of each vertex exists uniquely if it exists.) If a dicycle of length in $[3, s]$ is found, then it is rejected; otherwise, it is accepted.

We show that the above algorithm is the desired one-sided-error tester. It is clearly a one-sided-error, since it never rejects without finding a short (i.e., length of at most s) dicycle. Thus, it is enough to show that the algorithm rejects with probability at least $2/3$ if the input is ϵ-far from $\mathcal{P}_{C_{\sqrt{n}}}$.

Assume that the input $G = (V, E)$ is ϵ-far from $\mathcal{P}_{C_{\sqrt{n}}}$, i.e., that G contains more than ϵn dicycles of length in $[3, \sqrt{n}]$. Let C be the set of such dicycles. We divide C into the following two sets:

$C_{\text{short}} = \{C \in C \mid \text{the length of } C \text{ is at most } s.\}$

$C_{\text{long}} = \{C \in C \mid \text{the length of } C \text{ is more than } s \text{ (and at most } \sqrt{n}).\}$

From $|C| > \epsilon n$, $|C_{\text{short}}| > \epsilon n/2$ or $|C_{\text{long}}| > \epsilon n/2$ holds.

First, we assume that $|C_{\text{long}}| > \epsilon n/2$. Clearly no pair of dicycles in C shares a common vertex, and thus more than $\epsilon s n/2 = n$ vertices are included in the graph contradiction. Thus, $|C_{\text{long}}| \leq \epsilon n/2$.

From this, it follows that $|C_{\text{short}}| > \epsilon n/2$. Since no pair of dicycles in C shares a common vertex and each dicycle has at least three vertices, then the dicycles in C_{short} contain more than $3\epsilon n/2$ vertices. Let W be the set of such vertices. If the algorithm finds at least one vertex from W, then it will find a short dicycle that includes the vertex and rejects the input. From $|W| > 3\epsilon n/2$, it follows that the probability that a chosen vertex is not in W is less than $1 - 3\epsilon/2$. Thus, the probability that all of s vertices chosen by the algorithm are not in W is less than

$$(1 - 3\epsilon/2)^s \leq e^{-3\epsilon s/2} = e^{-3} < \frac{1}{3}.$$

Note that the first inequality above uses the inequality (2.4). The probability that the algorithm finds at least one vertex from W is, therefore, more than $2/3$. The query complexity of this tester is clearly $O(\epsilon^{-2})$. □

Since $\mathcal{P}_{C_{\sqrt{n}}}$ is both monotone and hereditary, Theorems 2.6 and 2.7 hold. If we apply Theorem 2.6, then $\mathcal{H}_n = C_{\min\{2/\epsilon, \sqrt{n}\}}$ for each n. If we apply Theorem 2.7, then $N(\epsilon) = 4/\epsilon^2$ and $\mathcal{H} = C_{2/\epsilon}$. From this discussion, we observe that $N(\epsilon)$ is essential in Theorem 2.7.

2.4.4 *An Idea to Extend the Characterizations Beyond Monotone and Hereditary*

We would like to extend Theorems 2.6 and 2.7 to general properties. We denote by $\mathcal{P}_{\deg^+(d-1)}$ the property consisting of digraphs having no vertex with out-degree $d - 1$ on the $F(d)$-model. $\mathcal{P}_{\deg(d-1)}$ is one-sided-error testable as shown below.

Let $G = (V, E)$ be an input. The algorithm for $\mathcal{P}_{\deg^+(d-1)}$ chooses $2/\epsilon$ vertices from V uniformly at random and checks their out-degrees. If it finds a vertex of degree $d-1$, then it is rejected; otherwise, it is accepted. This algorithm is a one-sided-error, since it never rejects if there is no vertex of out-degree $d-1$. If G is ϵ-far from $\mathcal{P}_{\deg^+(d-1)}$, then there are more than ϵn vertices of out-degrees $d-1$. Thus, the probability that there is no vertex of out-degree $d-1$ in the selected $2/\epsilon$ vertices by the algorithm is less than

$$(1-\epsilon)^{\frac{2}{\epsilon}} \leq \left(e^{-\epsilon}\right)^{\frac{2}{\epsilon}} = e^{-2} < \frac{1}{3}.$$

Note that this also uses the inequality (2.4).

Hence, the above algorithm is a one-sided-error tester for $\mathcal{P}_{\deg^+(d-1)}$. However, expressing this property by using forbidden subgraphs or forbidden induced subgraphs like Theorems 2.6 or 2.7 is impossible.[13]

To extend the idea of "forbidden something" to non-monotone and non-hereditary properties, we [12] introduced the idea of "configurations," by generalizing subgraphs and induced subgraphs. A similar idea has also appeared in [16].

Definition 2.9 A *configuration* is a pair $O = (H, L)$, where $H = (W, F)$ is a digraph in the $F(d)$-model, $L : W \to \{\text{developed}, \text{frontier}\}$ is a function, and the out-degree of every frontier vertex is 0. The configuration is *rooted* if H is rooted.

Definition 2.10 Let $O = (H = (W, F), L)$ and $G = (V, E)$ be a configuration and a graph respectively in the $F(d)$-model. We say that G has an *O-appearance* if there is an injective mapping $\phi : W \to V$ satisfying the condition that $\forall v \in W$ with $L(v) = \text{developed}$, the following two conditions hold:

(i) $\forall u \in W, (v, u) \in F$ if and only if $(\phi(v), \phi(u)) \in E$.
(ii) If $(\phi(v), x) \in E$, then $\exists u \in W, \phi(u) = x$.

We say that G is *O-free* if G has no O-appearance. For a set $\mathcal{O}$ of configurations, we say that G is *$\mathcal{O}$-free* if $\forall O \in \mathcal{O}$, G is O-free. □

As we have already stated, $\mathcal{P}_{\deg^+(d-1)}$ cannot be defined by any set of forbidden subgraphs or induced subgraphs. However, it can be defined by using $\mathcal{O}$-freeness. That is, let $O_{\deg^+(d-1)} = (H = (W, F), L)$ be a configuration such that $W = \{v_0, v_1, \ldots, v_{d-1}\}$, $E = \{(v_0, v_1), (v_0, v_2), \ldots, (v_0, v_{d-1})\}$, $L(v_0) = \text{developed}$, and $L(v_1) = L(v_2) = \cdots = L(v_{d-1}) = \text{frontier}$. Then $\mathcal{P}_{\deg(d-1)}$ is defined by the set of $O_{\deg(d-1)}$-free graphs.

The idea of configuration-free (or forbidden configurations) may work for characterizing general one-sided-error testable properties on the $F(d)$-model. See [12] for details.

[13] This follows from the fact that $\mathcal{P}_{\deg^+(d-1)}$ is neither monotone nor hereditary, and from Theorems 2.6, and 2.7.

2.5 Testable EXPTIME-Complete Games

This section presents results on the testability of combinatorial games, particularly the generalized chess, Shogi (Japanese chess), and Xiangqi (Chinese chess). Given any position on a $\sqrt{n} \times \sqrt{n}$ board with $O(n)$ pieces, the generalized chess, Shogi, and Xiangqi problems are the problems of determining the property that "the player who moves first has a winning strategy." These problems are known or believed to be EXPTIME-complete [1, 6, 7]. In [13], we proposed that this property is testable for chess, Shogi, and Xiangqi. The Shogi tester and Xiangqi tester are one-sided-error testers, and surprisingly, the chess tester is a no-error tester. Many problems have been revealed to be testable, but most of such problems belong to class NP. We think that this is the first result on the constant-time testability of EXPTIME-complete problems. This section presents these results. We mainly focus on chess, followed by Shogi, but omit the explanation for Xiangqi since the method is similar to the one for Shogi. See [13] for details.

2.5.1 Definitions

We begin by focusing mainly on generalized chess. Generalized chess is played on a $\sqrt{n} \times \sqrt{n}$ board with $O(n)$ pieces, including two kings. White moves first and black plays after white. A position is defined by fixing each piece to a particular cell on the board. At any given position S, the problem is to determine whether white wins if both players play optimally. The basic rules are the same as those in the original chess and are omitted here.

In chess, there are six different types of pieces: king (K), queen (Q), bishop (B), knight (N), rook (R), and pawn (P). There are only two pieces of kings; one white and one black. For each of the other piece-types (i.e., bishop, knight, rook, and pawn), there exist at most cn pieces for both white and black, respectively, where c is a constant. Piece-numbers from 1 to cn are given to each white or black piece of each piece-type; in other words, each piece has its own *piece ID* (k, o, ℓ) comprising a piece-type $k \in \{\mathrm{K}, \mathrm{Q}, \mathrm{B}, \mathrm{N}, \mathrm{R}, \mathrm{P}\}$, an owner-color $o \in \{\text{white}, \text{black}\}$, and a piece-number $\ell \in \{1, \ldots, cn\}$.

An algorithm can find the given position through the following oracles.

- *Piece oracle*: Given a piece ID (k, o, ℓ), the piece oracle answers an ordered pair (i, j) that provides the cell (i, j), $i, j \in \{0, 1, \ldots, \sqrt{n}\}$ where it lies. (i and j represent the column number and row number, respectively, and if $i = j = 0$, it denotes that the piece is not in the game (such a piece is called an *unused* piece). This oracle is expressed as $q_1(k, o, \ell) = (i, j)$.
- *Coordinate oracle*: Given a coordinate $(i, j), i, j \in \{0, 1, \ldots, \sqrt{n}\}$, the coordinate oracle answers the piece ID (k, o, ℓ) of the piece that lies on the cell if one exists. If no piece lies on the cell, the oracle answers $k = o = \ell = 0$. This oracle is expressed as $q_2(i, j) = (k, o, \ell)$.

When we explicitly identify position S, we express the oracles as $q_1(k, o, \ell; S)$ and $q_2(i, j; S)$, respectively. We introduce the assumption that all pieces can be arranged on the board simultaneously, and it thus follows that $2 \times (5cn + 1) \leq n$. For simplicity, we assume that

$$c \leq 1/11. \tag{2.5}$$

A position S is called a *winner* if white has a winning strategy (i.e., white will win if the players start from S and play optimally) and a *loser* otherwise. Note that a loser not only includes cases where white loses but also where the game ends in a draw.

A position is fixed by querying the piece oracle for every piece. The number of different queries for the piece oracle is at most n, and thus a position is fixed by the maximum of n data. From this, we define the *distance* between positions S and S' as

$$\mathrm{dist}(S, S') := \frac{|\{(i, j) \mid q_2(i, j; S) \neq q_2(i, j; S')\}|}{n}. \tag{2.6}$$

Clearly $0 \leq \mathrm{dist}(S, S') \leq 1$.

Positions S and S' are called *isomorphic* if we can make S identical to S' by only changing their piece-numbers (neither changing the piece-type nor owner-color is allowed). A set of positions that is closed under isomorphism is called a *property*. The distance between a position S and a property $\mathcal{P}$ is defined as follows:

$$\mathrm{dist}(S, \mathcal{P}) := \min_{S' \in \mathcal{P}} \mathrm{dist}(S, S'). \tag{2.7}$$

For a positive $\epsilon > 0$, S is *ϵ-far* from $\mathcal{P}$ if $\mathrm{dist}(S, \mathcal{P}) > \epsilon$; otherwise, it is *$\epsilon$-close*. Let $\mathcal{W}$ be the set of winners. $\mathcal{W}$ is clearly closed under isomorphism and thus $\mathcal{W}$ is a property.

For generalized Shogi and Xiangqi, similar definitions are used. They can be easily deduced and are omitted here. See [13] for details.

2.5.2 Testers for Generalized Chess, Shogi, and Xiangqi

The following theorem was presented in [13]. Note that a *no-error* tester is a one-sided-error tester that always rejects every input that is ϵ-far from the property; that is, it always accepts or rejects with no-error if the input is in the property or ϵ-far from the property.

Theorem 2.8 *There exists a no-error tester with query complexity $O(\epsilon^{-1})$ for the generalized chess problem, there exists a one-sided-error tester with query complexity $O(\epsilon^{-2})$ for the generalized Shogi problem, and there exists a one-sided-error tester with query complexity $O(\epsilon^{-1})$ for the generalized Xiangqi problem.*

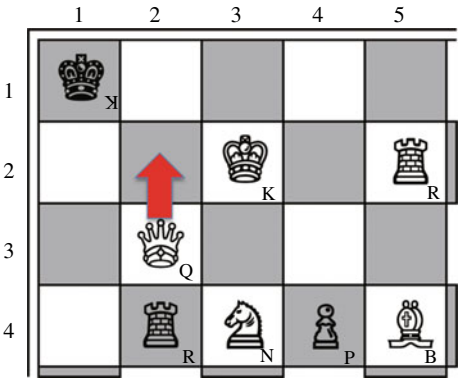

Fig. 2.1 The black king will be checkmated by white's next move, as indicated by the arrow

Proof of the chess part of Theorem 2.8 Let S be a given position. Let S' be the position made from S by changing the pieces in cells (i, j), $i \in \{1, 2, 3, 4\}$ and $j \in \{1, 2, 3, 4, 5\}$, as shown in Fig. 2.1.

The pieces that were in these cells in S are changed to be unused pieces, and the pieces that appear in these cells in S' are moved from other cells or unused pieces. In S', the white king is safe and the black king will be checkmated by white's next move (moving the queen from $(3, 2)$ to $(2, 2)$), meaning that S' is a winner. The distance between S and S' is at most $20 + 8 = 28$. Thus, if $n \geq 28/\epsilon$, then $\mathrm{dist}(S, S') \leq 28/n \leq \epsilon$. Hence, S is ϵ-close to $\mathcal{W}$, and it is sufficient to accept it. If $n < 28/\epsilon$, it is sufficient to read all of the information by calling the piece oracle for all pieces, which requires $O(\epsilon^{-1})$ queries.

This algorithm always accepts a winner. Moreover, if a given position S is ϵ-far from $\mathcal{W}$, then $n < 28/\epsilon$ and the algorithm knows the complete information for S. Therefore, this algorithm is no-error. □

The algorithms for the generalized Shogi and Xiangqi problems are a little more complicated. The reason is that in Shogi and Xiangqi there are fouls based on positions. A player who plays the fouls loses. In Shogi, the following fouls need to be considered in the generalized Shogi problem.

- *Nifu (double pawn)*: two or more unpromoted[14] pawns that belong to the same player must not be in the same column simultaneously.
- *Dead end*: pawns, lances, and knights[15] can never be moved or dropped onto cells from which a subsequent move cannot be made. Therefore, white (resp., black) unpromoted pawns and lances can never be in the first (resp., $\sqrt{n}$th) row, and white (resp., black) knights can never be in the first or second (resp., $\sqrt{n}$th or $(\sqrt{n} - 1)$th) rows.

[14] If a piece of some piece-type can be promoted (to a stronger piece) when it enters the opponent's camp.

[15] These three pieces can move only forward.

In a given position S, if there is a white piece that plays a fault, then white cannot win,[16] and thus the position is not a winner. However, if the number of pieces related to fouls is small (e.g., smaller than $\epsilon n/2$), we can remove the fouls and make white win, i.e., the position is ϵ-close to $\mathcal{W}$. To detect this, we need to perform preprocessing and the tester may have error when the input is ϵ-far from $\mathcal{W}$. For Xiangqi, a similar discussion applies. See [13] for details.

2.6 Summary

In this chapter, we introduced basic terminology and important results for property testing, which is the most examined framework for constant- or sublinear-time algorithms. In particular we presented two of our resent esults: The first is the complete characterization of one-sided-error testable monotone or hereditary properties on bounded-out-degree digraphs, and the other one is the testers for the generalized chess, Shogi, and Xiangqi problems, which are all EXPTIME-complete.

The 21st century can be called the era of big data, and the larger big data becomes, the more we need sublinear- and constant-time algorithms. The importance of this area will continue to grow. The number of fields in which constant-algorithms are efficiently applied will increase, and new techniques will be found accordingly. We eagerly await these developments.

References

1. H. Adachi, H. Kamekawa, S. Iwata, Shogi on $n \times n$ board is complete in exponential time. IEICE J. **J70-D**(10), 1843–1852 (1987). (In Japanese)
2. N. Alon, E. Fischer, I. Newman, A. Shapira, A combinatorial characterization of the testable graph properties: it's all about regularity. SIAM J. Comput. **39**(1), 143–167 (2009)
3. J. Babu, A. Khoury, I. Newma, Every property of outerplanar graphs is testable, in *Proceedings of the RANDOM 2016*, LIPICS, pp. 21:1–21:19 (2016)
4. A. Bhattacharyya, Y. Yoshida, *Property Testing: Problems and Techniques* (Springer, Berlin, 2021) (scheduled to be published in 2021)
5. H. Fichtenberger, P. Peng, C. Sohler, Every Testable (Infinite) Property of Bounded-Degree Graphs Contains an Infinite Hyperfinite Subproperty (2018). arXiv: 1811.02937 (also appeared in SODA2019)
6. R.H. Fleischer, S.U. Khan, Xiangqi and combinatorial game (2002)
7. A.S. Fraenkel, D. Lichtenstein, Computing a perfect strategy of $n \times n$ chess requires time exponential in n. J. Comb. Theory Ser. A **31**, 199–214 (1981)
8. O. Goldreich, *Introduction to Property Testing* (Cambridge University Press, Cambridge, 2017)
9. H. Ito, K. Iwama, Enumeration of isolated cliques and pseudo-cliques. ACM Trans. Algorithms **5**(4), Article 40, 1–13 (2009)
10. H. Ito, Every property is testable on a natural class of scale-free multigraphs, in *Proceedings of ESA 2016*, LIPICS, Vol. 49 (ISBN 978-3-95977-005-7) (2016), pp. 21:2–21:15

[16] If both players play fouls, the game is a draw.

11. H. Ito, What graph properties are constant-time testable?—dense graphs, sparse graphs, and complex networks. The Rev Socionetw Strat, Springer **13**(2), 101–121 (2019)
12. H. Ito, A. Khoury, I. Newman, On the characterization of 1-sided error strongly-testable graph properties for bounded-degree graphs. J. Comput. Compl. Springer **29**, Article Number 1, 1–45 (2020)
13. H. Ito, A. Nagao, T. Park, Generalized shogi, chess, and xiangqui are constant-time testable. IEICE Trans. **E102-A**(9), 1126–1133 (2019)
14. M. Kusumoto, Y. Yoshida, Testing forest-isomorphizm in the adjacency list model, in *Proceedings of ICALP2014* (1), LNSC 8572 (2014), pp. 763–774
15. I. Newman, C. Sohler, Every property of hyperfinite graphs is testable, in *Proceedings STOC 2011* (ACM, 2011), pp. 675–784. (Journal version: SIAM J. Comput. vol. 42, No. 3, pp. 1095–1112 (2013).)
16. Oded Goldreich, Dana Ron, Property testing in bounded degree graphs. Algorithmica **32**(2), 302–343 (2002)
17. E. Szemerédi, Regular partitions of graphs, in Proceedings of the Colloquim International CNRS, eds. by J.C. Bermond, J.C. Fournier, M. Las Vergnas, D. Sotteau (CNRS, Paris, 1978), pp. 399–401