Testing Graph Properties

Artur Czumaj
DIMAP (Centre for Discrete Maths and it Applications) and
Department of Computer Science
University of Warwick
Main topic

• How to test basic properties of graphs in the framework of property testing
Property Testing

- Distinguish inputs that have specific property from those that are far from having the property.
Property testing

• Classical decision problem:
  – Given a property $P$ and input instance $I$
  – Does $I$ has property $P$?

Often it’s computationally hard (NP-complete/undecidable)

• What we want to study [relaxation]:
  – Is $I$ close to satisfy property $P$?

Can work fast even for NP-hard or undecidable properties
Property Testing definition

- Given input x
- If x has the property $\Rightarrow$ tester passes
- If x is $\varepsilon$-far from any string that has the property $\Rightarrow$ tester fails
- error probability < $1/3$

Notion of $\varepsilon$-far depends on the problem; Typically: one needs to change $\varepsilon$ fraction of the input to obtain an object satisfying the property

Typically we think about $\varepsilon$ as on a small constant, say, $\varepsilon = 0.1$
Property Testing definition

- Given input $x$
- If $x$ has the property $\Rightarrow$ tester passes
- If $x$ is $\varepsilon$-far from any string that has the property $\Rightarrow$ tester fails
- error probability $< 1/3$

- This is two-sided error tester
- one-sided error: errs only for $x$ being $\varepsilon$-far
So, what is property testing

- Early motivation:
  - Program checking
  - Program verification
  - Learning theory

- Relation to Probabilistically Checkable Proofs
Properties of functions

- **Linearity test** [Blum Luby Rubinfeld] [Bellare Coppersmith Hastad Kiwi Sudan] (various improvements by many others)
  \[
  \forall x, y \ f(x)+f(y)=f(x+y)
  \]
- **Low total degree polynomial tests** [Rubinfeld Sudan] [Arora Safra] [Arora Lund Motwani Sudan Szegedy] [Arora Sudan] ...
- **Functions definable by functional equations** – trigonometric, elliptic functions
- **Groups, Fields**
- **Finite precision** [Gemmell Lipton Rubinfeld Sudan Wigderson] .......
- **Low complexity functions** [Parnas Ron Samorodnitsky] ..... 

- **Useful in**
  - Program checking
  - PCP constructions
Study of combinatorial properties
[Goldreich Goldwasser Ron’98]

- Graph properties
- Hypergraph properties
- Monotonicity
- Set properties
- Geometric properties
- String properties
- Membership in low complexity languages
  (regular languages, constant width branching programs, context-free languages ...
The focus of this talk: Properties of graphs

- Graph properties [Goldwasser, Goldreich, Ron’98]:
  - Colorability
  - Not containing a forbidden subgraph
  - Not containing a forbidden minor
  - Connectivity
  - Acyclicity
  - Rapid mixing
  - Expansion
  - Max-Cut
  ...

Some of these properties are NP-hard
Goal:
Distinguish between the case when
- graph G has property P and
- G is far from having property P

• **one has to change G in an $\varepsilon$ fraction of its representation to obtain a graph with property P**

• What does it mean “an $\varepsilon$ fraction of its representation”?
1st definition

Adjacency Matrix

Graph G is \( \varepsilon \)-far from satisfying property P

If one needs to modify more than \( \varepsilon \)-fraction of entries in adjacency matrix to obtain a graph satisfying P

Access to G via oracle:

is i connected by edge to j? 

\((A[i, j] = 1?)\)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Graph G is \( \varepsilon \)-far from satisfying property P

If one needs to modify more than \( \varepsilon \)-fraction of entries in adjacency matrix to obtain a graph satisfying P

- \( \varepsilon \cdot n^2 \) edges have to be added/deleted
- Suitable for dense graphs
- Usually “boring” for sparse graphs
Graph G is $\varepsilon$-far from satisfying property P
If one needs to modify more than $\varepsilon$-fraction of entries in adjacency lists to obtain a graph satisfying P

Access to G via oracle:
Return the $i$th neighbor of $v$
Graph $G$ is $\varepsilon$-far from satisfying property $P$

If one needs to modify more than $\varepsilon$-fraction of entries in adjacency lists to obtain a graph satisfying $P$

$\varepsilon \cdot |E|$ edges have to be added/deleted

Suitable for sparse graphs
Graph G is \( \varepsilon \)-far from satisfying property P

If one needs to modify more than \( \varepsilon \)-fraction of entries in adjacency lists to obtain a graph satisfying P.

\[ \varepsilon \cdot |E| \text{ edges have to be added/deleted} \]

Main model: graphs of bounded degree

\[ \varepsilon \cdot d \cdot n \text{ edges have to be added/deleted} \]
Plan

1) Adjacency matrix model
   - Examples
   - Characterization of testable properties

2) Adjacency list model
   - Basic examples
   - Attempts to characterize complexity of testing
     - Bounded-degree arbitrary graphs
     - Special classes (e.g., planar) of bounded-degree graphs
     - Graphs with arbitrary degrees
Adjacency matrix model

- **Accept** every graph that satisfies property $P$
- **Reject** every graph that is $\epsilon$-far from property $P$
  - $\epsilon$-far from $P$: one has to modify at least $\epsilon n^2$ entries of the adjacency matrix to obtain a graph with property $P$
- **Arbitrary answer** if the graph doesn’t satisfy $P$ nor is $\epsilon$-far from $P$
- **Complexity**: number of queries to the matrix entries
- Can err with probability $< 1/3$
  - Sometimes errs only for “rejects”: one-sided-error

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Adjacency matrix model

Very easy example:
• Test if a graph contains a triangle (cycle of length 3)

Return YES (always)

Highly nontrivial example:
• Test if a graph is triangle-free

• Can be done in $f(\varepsilon) = O(1)$ time
• Proof: deep combinatorics


Wlog [Goldreich, Trevisan’03] we can consider only algorithms of the following form: Any other algorithm will have not more than a quadratic speed-up

Randomly sample set S of vertices
Consider subgraph of G induced by S
If the subgraph satisfies a property $\Rightarrow$ accept
otherwise $\Rightarrow$ reject

This property may be **different** from the original one
Adjacency matrix model

- Test if a graph $G=(V,E)$ is planar
  - We’ve already said: testing is “trivial” for sparse graphs ...

- If $G$ is dense then it’s certainly not planar
- If $G$ is sparse then it’s not $\varepsilon$-far from planar
  - Every graph with less than $\varepsilon n^2/2$ edges is $\varepsilon$-close to planar: remove all its edges

- How to distinguish between sparse and dense graphs?
  - test $O(1/\varepsilon)$ entries in the matrix: if all zeros then sparse
- Two-sided error tester with complexity $O(1/\varepsilon)$
There are very fast property testers
They’re very simple
Property tester for bipartitiveness:

- Select a random set of vertices $U$
- Test if the subgraph induced by $U$ is bipartite

Key question: What should be the size of $|U|$?

- Goldreich, Goldwasser, Ron: $|U| = \text{poly}(1/\varepsilon)$
- Alon, Krivelevich: $|U| = O^*(1/\varepsilon) \Rightarrow \text{complexity } O^*(1/\varepsilon^2)$
General result

• Every hereditary property can be tested in **constant-time**! (even with one-sided error)

  [Alon & Shapira, 2003-2005]

• Property is **hereditary** if
  - It holds if we remove vertices
    • bipartitness
    • being perfect
    • being chordal
    • having no induced subgraph H
    • ...

• Connection to Szemeredi lemma
Main Lemma:

If $G$ is $\varepsilon$-far from satisfying a hereditary property $P$, then whp random subgraph of size $W_p(\varepsilon)$ doesn’t satisfy $P$

Proof: by a strengthened version of Szemeredi regularity lemma

Can be extended to hypergraphs

- via a strengthened version of Szemeredi regularity lemma for hypergraphs
• Every hereditary property can be tested in **constant-time!**
  (even with one-sided error)
  
  [Alon & Shapira, 2003-2005]

• Being hereditary is essentially necessary and sufficient for one-sided error

**Almost complete characterization of graph properties testable in constant-time with one-sided error**
General result

• Every hereditary property can be tested in constant-time! (even with one-sided error)
  [Alon & Shapira, 2003-2005]

• Similar characterization for two-sided error testing
  Informally:
  A graph property is testable in constant-time iff testing can be reduced to testing finitely many Szemeredi partitions
  [Alon, Fischer, Newman, Shapira’09]
Adjacency matrix model

- There are very fast property testers
- They’re very simple
  - Typical algorithm:
    - Select a random set of vertices U
    - Test the property on the subgraph induced by U

- The analysis is (often) very hard
- We understand this model very well
  - mostly because of very close relation to combinatorics
  - Typical running time: (via Szemeredi regularity lemma)
Adjacency matrix model

- There are very fast property testers
- They’re very simple
  - Typical algorithm:
    - Select a random set of vertices $U$
    - Test the property on the subgraph induced by $U$
- The analysis is (often) very hard
- We understand this model very well
  - mostly because of very close relation to combinatorics
  - Typical running time: (via Szemeredi regularity lemma)

$$\text{Tower}(c) = 2^{2^{\ldots^{2^c}}}, \text{ } c \text{ times}$$
Adjacency matrix model

- There are very fast property testers
- They’re very simple
  - Typical algorithm:
    - Select a random set of vertices U
    - Test the property on the subgraph induced by U
- The analysis is (often) very hard
- We understand this model very well
  - mostly because of very close relation to combinatorics
  - Typical running time: (via Szemeredi regularity lemma)

\[ \text{Tower(Tower(Tower(1/\varepsilon))))} \]

For \(\varepsilon=0.5\) we have \(\text{Tower(Tower(Tower(1/\varepsilon)))) = \text{Tower(65536)}\)
Adjacency matrix model

- There are very fast property testers
- They’re very simple
  - Typical algorithm:
    - Select a random set of vertices U
    - Test the property on the subgraph induced by U
- The analysis is (often) very hard
- We understand this model very well
  - mostly because of very close relation to combinatorics
- Still: sometimes the runtime is better
  \( O(1/\varepsilon), O(1/\varepsilon^2), O(1/2^\varepsilon) \)
What’s about adjacency lists model?

- We consider bounded-degree model
  - graph has maximum degree $d$ [constant]

- Much less is known

- Less connection to combinatorics

- Main techniques:
  - random sampling
  - local search (exploring the neighborhood/ball of a vertex)
  - random walks (a random neighbor of a random neighbor of a random neighbor...)

Bounded-degree adjacency list model

- **Accept** every graph of max-degree \(d\) that satisfies property \(P\)
- **Reject** every graph of max-degree \(d\) that is \(\varepsilon\)-far from property \(P\)
  - \(\varepsilon\)-far from \(P\): one has to modify at least \(\varepsilon dn/2\) edges to obtain a graph with property \(P\)
- **Arbitrary answer** if the graph doesn’t satisfy \(P\) nor is \(\varepsilon\)-far from \(P\)
- Can err with probability < 1/3
  - Sometimes errs only for “rejects”: one-sided-error
Bounded-degree adjacency list model

Testing connectivity

What does it mean that a graph $G$ with maximum degree at most $d$ is $\varepsilon$-far from connected?

$\Rightarrow G$ has at least $\varepsilon dn/8$ connected components

- not enough...we need many small connected components
What does it mean that a graph $G$ with maximum degree at most $d$ is $\varepsilon$-far from connected?

$G$ has $\geq \varepsilon dn/16$ connected components of size $\leq 16/\varepsilon d$

Repeat $O(\varepsilon^{-1} d)$ times:
- choose a random vertex $v$
- run BFS from $v$ until either $16/\varepsilon d + 1$ vertices have been visited or the entire connected component has been visited
- if $v$ is contained in a connected component of size $\leq 16/\varepsilon d$ then reject
- accept

Testing connectivity can be done in $O(\varepsilon^{-2} d)$ time
Testing connectivity was easy ...
Let’s move to the real stuff

What properties can be tested in constant-time?
We want a characterization!
Bounded-degree adjacency list model

• Testing bipartitness
  – Can be done in \( O^*(n^{1/2} / \varepsilon^{O(1)}) \) time (Goldreich & Ron)

Algorithm:
• Select \( O(1/\varepsilon) \) starting vertices
• For each vertex run \( \text{poly}(\varepsilon^{-1} \log n) n^{1/2} \) random walks of length \( \text{poly}(\varepsilon^{-1} \log n) \)
• If any of the starting vertices lies on an odd-length cycle then reject
• Otherwise accept

Idea:
• if \( G \) is \( \varepsilon \)-far from bipartite then \( G \) has many odd-length cycles of length \( O(\varepsilon^{-1} \log n) \)
• run many short random walks to find one
Bounded-degree adjacency list model

- Testing bipartitiveness
  - Can be done in \( O^*(n^{1/2} / \varepsilon^{O(1)}) \) time (Goldreich & Ron)

**Algorithm:**
- Select \( O(1/\varepsilon) \) starting vertices
- For each vertex run \( \text{poly}(\varepsilon^{-1} \log n) n^{1/2} \) random walks of length \( \text{poly}(\varepsilon^{-1} \log n) \)
- If any of the starting vertices lies on an odd-length cycle then reject
- Otherwise accept

**Analysis:** very elaborate
- relatively easy for rapidly mixing case
- for general case: no rapid mixing \( \Rightarrow \) small cut
  use small cut to decompose the graph and the problem
• Testing bipartiteness
  – Can be done in $\mathcal{O}^*(n^{1/2}/\epsilon^{O(1)})$ time (Goldreich & Ron)
  – Cannot be done faster (Goldreich & Ron)

$\Omega(\sqrt{n})$ time is needed to distinguish between random graphs from the following two classes:

• Hamiltonian cycle $H +$ a perfect matching

• Hamiltonian cycle $H +$ a perfect matching $M$ such that each edge from $M$ creates an even-length cycle when added to $H$
Bounded-degree adjacency list model

- Testing bipartitness
  - Can be done in $O^*(n^{1/2} / \varepsilon^{0(1)})$ time (Goldreich & Ron)
  - Cannot be done faster (Goldreich & Ron)

- So: no constant-time algorithms
Bounded-degree adjacency list model

• Testing 3-colorability

... requires checking (almost) all vertices and edges!
Bounded-degree adjacency list model

Testing cycle-freeness (acyclicity)

Complexity depend on the error-model

- One-sided error (always accept cycle-free graphs)
- Two-sided error (can err for acceptance and rejection)
Testing cycle-freeness in bounded-degree graphs

two-sided error

Testing cycle-freeness

Complexity depend on the error-model

• One-sided error (always accept cycle-free graphs)

• Two-sided error (can err for acceptance and rejection)

Can be done with $O(\varepsilon^{-2}(d+\varepsilon^{-1}))$ samples

Goldreich and Ron’02:

Estimate the number of edges
Estimate the number of connected components
If these number are OK for a forest then accept
Else reject
Testing cycle-freeness in bounded-degree graphs

one-sided error

Testing cycle-freeness

Complexity depend on the error-model

• **One-sided error** (always accept cycle-free graphs)
• **Two-sided error** (can err for acceptance and rejection)

Goldreich, Ron’02:
A lower bound of $\Omega(n^{1/2})$
Testing cycle-freeness in bounded-degree graphs

one-sided error

Testing cycle-freeness

Complexity depend on the error-model

• One-sided error (always accept cycle-free graphs)
• Two-sided error (can err for acceptance and rejection)

C, Goldreich, Ron, Seshadhri, Sohler, Shapira ‘10
An upper bound of $O^*(n^{1/2})$:
reduction to bipartitness
C, Goldreich, Ron, Seshadhri, Sohler, Shapira ’10

Testing cycle-freeness can be done in $O^*(n^{1/2})$:

- we know how to test bipartitiveness (no odd-length cycles)
- reduce testing cycle-freeness to that of bipartitiveness
Idea: original graph $G$ has lots of cycles iff new graph $G^*$ has lots of odd-length cycles.
Put a new node on some edges ...
(some = random half)
If the original graph is cycle-free then the obtained graph is bipartite.

With high probability:
original graph is $\varepsilon$-far from cycle-free $\Leftrightarrow$
obtained graph is $\Theta(\varepsilon)$-far from bipartite.
Testing cycle-freeness in bounded-degree graphs
one-sided error

C, Goldreich, Ron, Seshadhri, Sohler, Shapira ’10

Testing cycle-freeness can be done in $O^*(n^{1/2})$:

- we know how to test bipartitness (no odd-length cycles)
- reduce testing cycle-freeness to that of bipartitiveness

Since testing bipartitiveness can be done in $O^*(n^{1/2})$ time, so does testing of cycle-freeness
Property:

**Being $C_k$-minor free** (having no cycle of length $\geq k$)

For every constant $k$, testing if a bounded-degree graph $G$ is $C_k$-minor free can be done in $O^*(n^{1/2})$
Bounded-degree adjacency list model

C, Goldreich, Ron, Seshadhri, Sohler, Shapira ’10

For every constant k, testing if a bounded-degree graph is $C_k$-minor free can be done in $O^*(n^{1/2})$

Can we do better?

For any fixed H that contains a simple cycle, testing minor $H$-freeness with one-sided error requires $\Omega(n^{1/2})$ time

Goldreich, Ron’02 proved it for $H=C_2$
Bounded-degree adjacency list model

C, Goldreich, Ron, Seshadhri, Sohler, Shapira ’10

For every constant $k$, testing if a bounded-degree graph is $C_k$-minor free can be done in $O^*(n^{1/2})$

For any fixed $H$ that contains a simple cycle, testing minor $H$-freeness with one-sided error requires $\Omega(n^{1/2})$ time

For any fixed tree $T$, testing minor $T$-freeness with one-sided error can be done in constant-time
Characterization of testing H-minor freeness
(in bounded-degree graphs, with one-sided error):

For any fixed H, testing if a graph is H-minor-free can be
done in complexity that only depends on $\epsilon$
if and only if
H is cycle-free

Testing H-minor-freeness for H having a cycle
needs time $\Omega(n^{1/2})$, but we don’t have any
further good complexity characterization
What’s about planar graphs?

• There are graphs $G$ such that
  – any connected subgraph of $G$ of constant size is planar
  – $G$ is $\varepsilon$-far from planar

For example, there’re bounded-degree expanders on $n$ nodes with $\omega(1)$ girth

So: testing any constant-size subgraph won’t give us a tester $\Omega(n^{1/2})$ lower bound for one-sided error planarity testing
Testing planarity

Testing planar graphs can be done with $O(1)$ queries (with two-sided error) [Benjamini, Schramm, Shapira’08]

- Why is it surprising?
- There are graphs $G$ such that
  - any connected subgraph of $G$ of constant size is planar
  - $G$ is $\varepsilon$-far from planar

For each subgraph of constant size, check the number of its occurrences in $G$

No all frequencies are possible in planar graphs!
Checking planarity in constant time

Randomly sample \(O(1)\) vertices
For each sampled vertex \(v\)
  run BFS from \(v\) of \(O(1)\) depth
Estimate how often each constant-size graph appears in \(G\)
Accept or reject \(G\) using estimations of the frequencies
Testing planar graphs can be done with $O(1)$ queries \cite{Benjamini, Schramm, Shapira'08}

- Runtime: $2^{2^{\text{poly}(1/\varepsilon)}}$

- Hassidim, Kelner, Nguyen, Onak’09 improved the runtime to $2^{\text{poly}(1/\varepsilon)}$
  - with somewhat simpler analysis and simpler algorithm
  - using the concept of a partitioning oracle

If $G$ is $\varepsilon$-far from planar then
- either $G$ has lots of constant-size non-planar subgraphs
- or $G$ has lots of small subgraphs without good separator
Testing planarity

Testing planar graphs can be done with $O(1)$ queries

[Benjamini, Schramm, Shapira’08]

[Hassidim, Kelner, Nguyen, Onak’09]

• Runtime: $2^{\text{poly}(1/\varepsilon)}$

• The result is with two-sided error:
  – can accept non-planar graphs & can reject planar graphs

There is no $o(n^{1/2})$-time one-sided-error tester for planarity
Extension: all minor-closed properties

- Every minor-closed property can be tested in a similar way.
- Minor-closed properties include:
  - Planar,
  - Outer-planar,
  - Series-parallel,
  - Bounded-genus,
  - bounded tree-width,
  - ...

- Minor = obtained by edge/vertex removal + edge contractions
- P is minor-closed if every minor of a graph in P is also in P.
Summary: Graph Property Testing

- We understand graph properties better and better
- Adjacency matrix model:
  - Complete characterization
- Adjacency lists model:
  - Bounded-degree graphs
    - Some basic characterizations known
    - Many open questions still left
  - Special classes of bounded-degree graphs
    - For “non-expanding” graphs we can test efficiently
  - Graphs with no bounds for the degree
    - Almost nothing is known; hard
Final comments

• A few years ago, after the characterization of constant-time testable properties for dense graphs due to Alon and Shapira, I thought no interesting problems in property testing left

• Now: lots of new results: using combinatorics + local search + analysis of random walks (and similar randomized processes)

• Still lots of open problems left
Future of Property Testing

We need general results

- characterization of testable properties
- characterization of properties testable within given time-bound
Future of Property Testing

Implications for:

- **approximation algorithms**
  - $O(1)$-time algorithms approximating cost of an MST, matching, minimum vertex cover, ...

- **distributed algorithms**
  - $O(1)$-distributed algorithms finding almost optimal matching, max-independent set, etc, in planar graphs of bounded-degree

- **streaming**