Understanding the Structure of Massive Graphs through the Lens of Property Testing

Artur Czumaj

Department of Computer Science

Centre for Discrete Mathematics and its Applications (DIMAP)

University of Warwick

Workshop on Algorithms for Massive Graphs (AMG 2024)
Madrid, Spain, November 1, 2024

Lens of Property Testing

Leading theme in Distributed Computing:

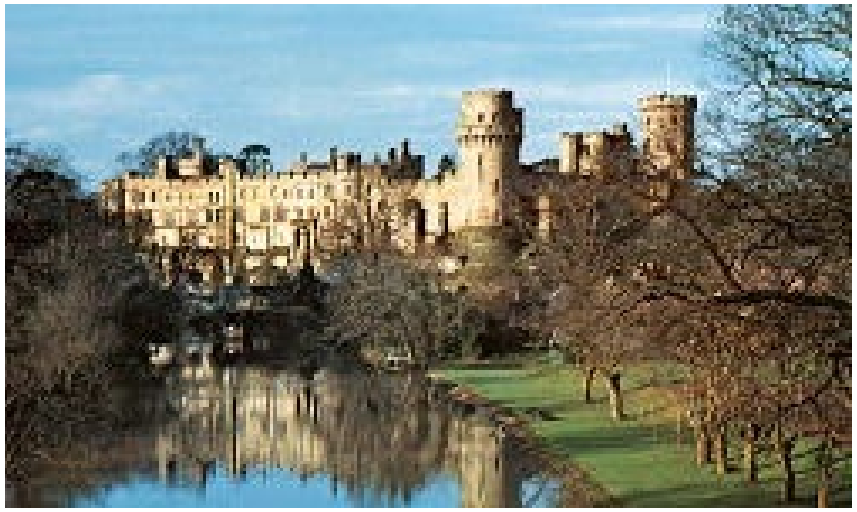
fight between LOCAL and GLOBAL

Property testing is a viewpoint trying to formalize some aspects of this trade-off: what can we learn if we see only a small sample of the input

Lens of Property Testing

This talk:

- Introduction to the area of Property Testing
- Focus on Graphs
- If there will be enough time:
  - some connections between property testing and other settings



Property Testing

- Distinguish inputs that have specific property from those that are far from having the property (Rubinfeld-Sudan'96)

Benefits:

- May be a natural question to ask
- May be just as good when data constantly changing
- Gives fast sanity check to rule out very "bad" inputs (i.e., restaurant bills) or to decide when expensive processing is worth it

Property Testing

Classical decision problem:

- Given a property P and input instance I
- Does I has property P?

Often it's hard (NP-complete or even undecidable)

What we want to study [relaxation]:

- Is I close to satisfying property P?

Can work fast even for NP-hard or undecidable properties

Property Testing definition

- Given input x
- If x has the property P ⇨ tester passes
- If x is ε-far from any string that has the property P ⇨ tester fails

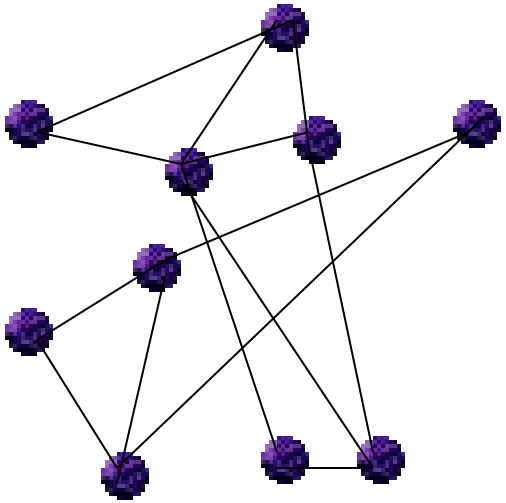
error probability < 1/3

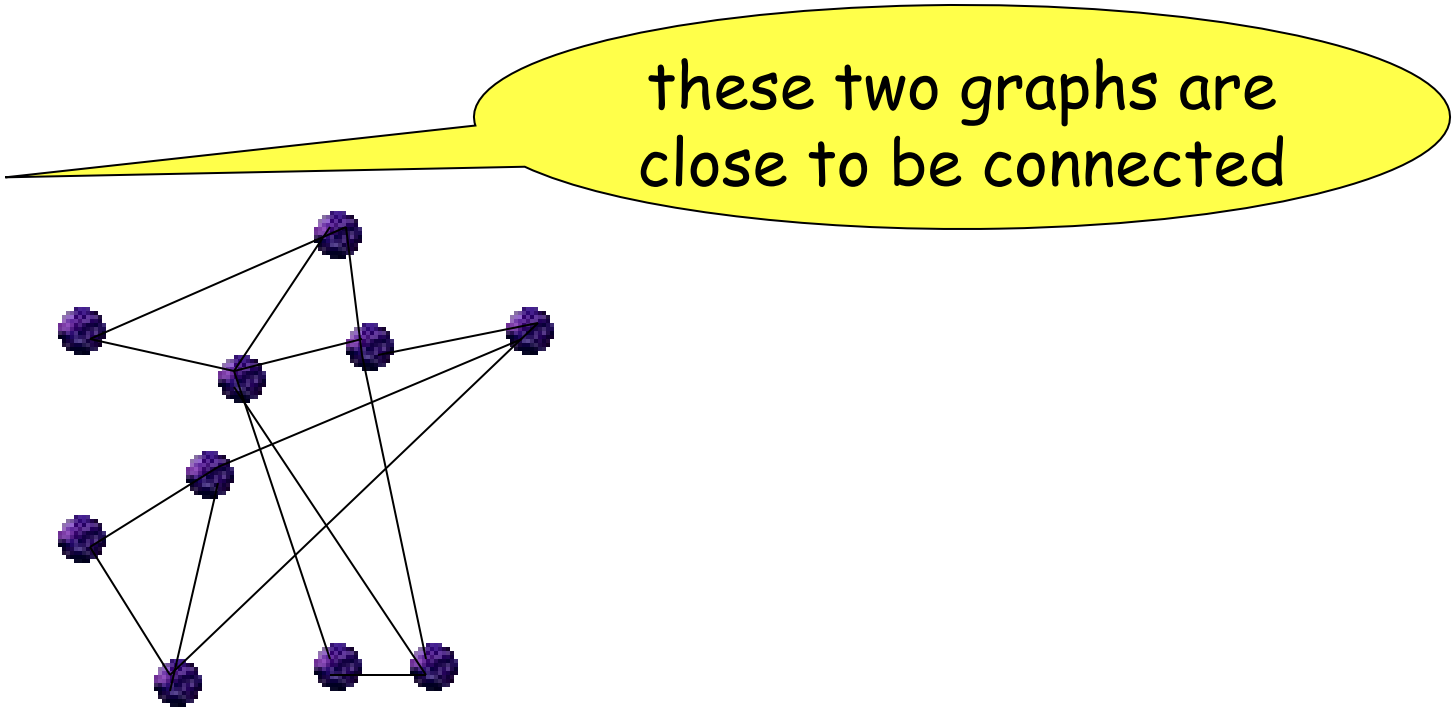
Notion of **ε-far** depends on the problem.
Typically: one needs to change an ε-fraction of the input to obtain object satisfying the property

Often we think about ε as on a small constant, say, **$\varepsilon = 0.1$**

Graph properties

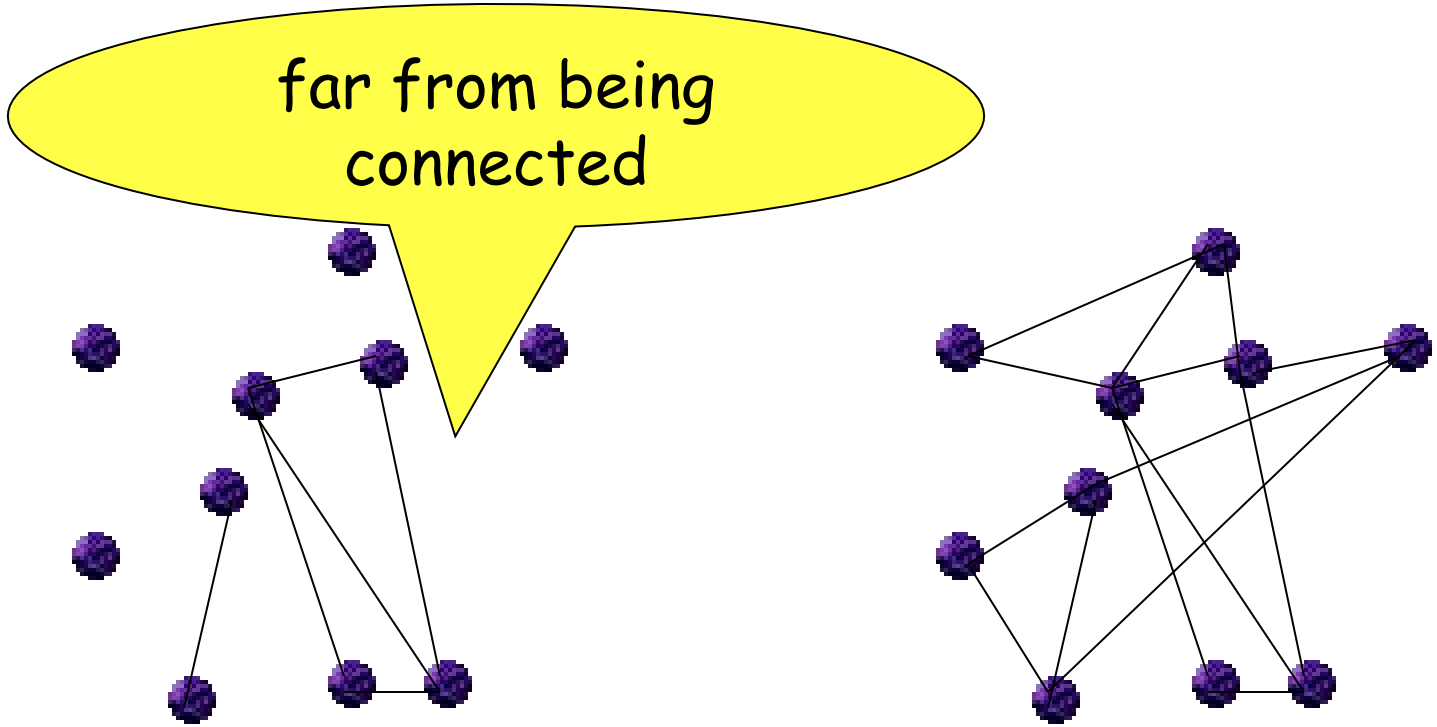
- Measure of being far/close from a property
- Is graph connected or is *far* from being connected?



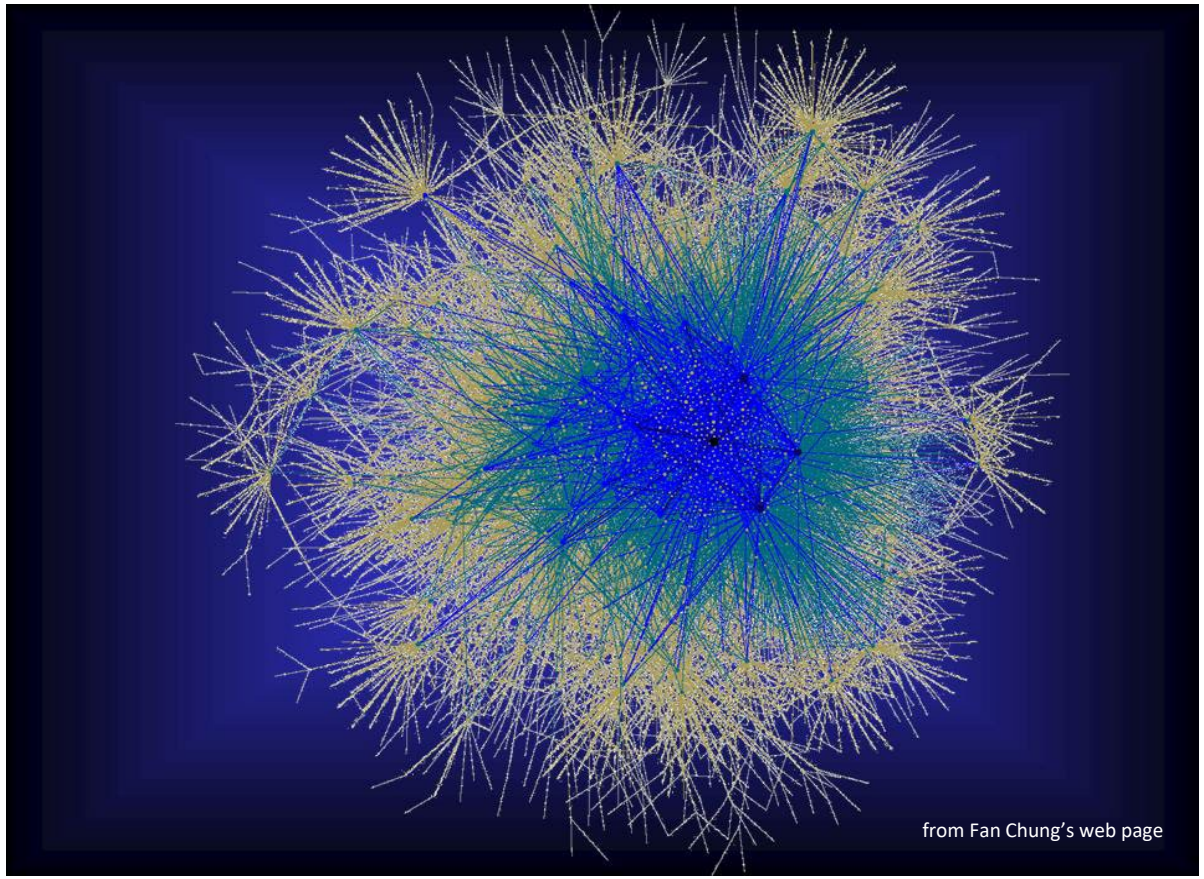


Graph properties

- Measure of being far/close from a property
- Is graph connected or is *far* from being connected?



Testing of Graph Properties

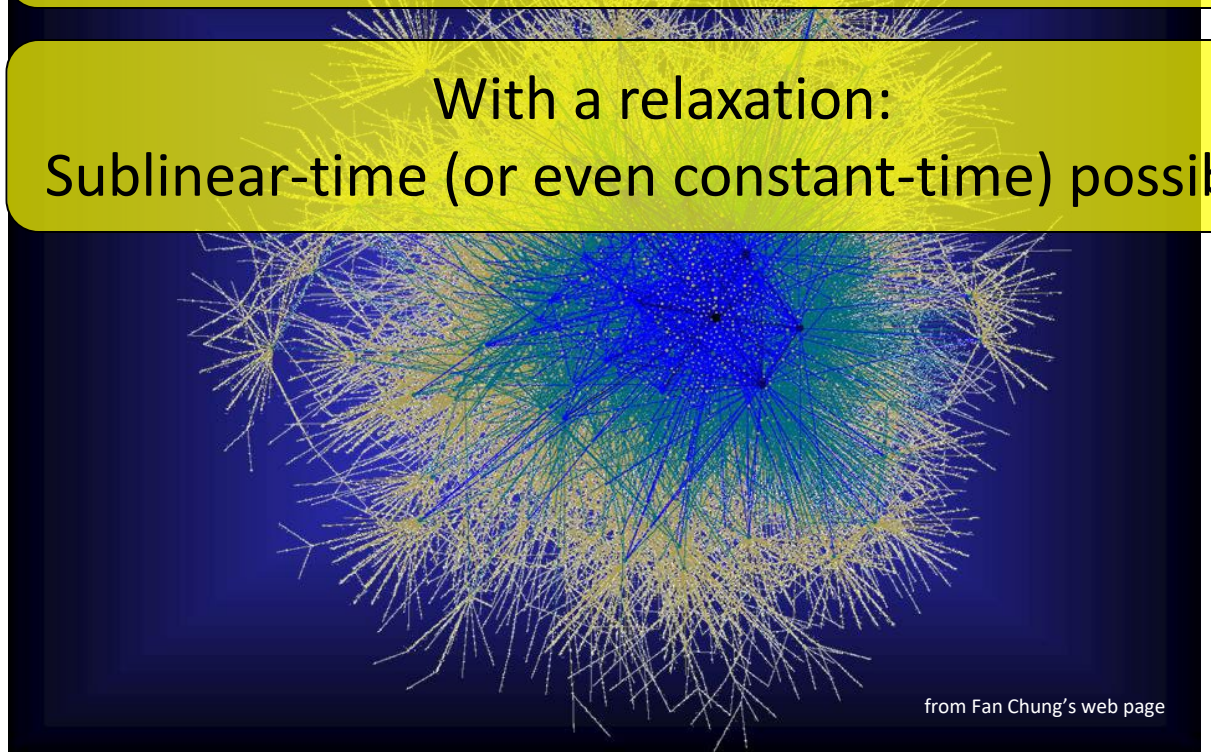


- Does this graph have a clique of size 11?
- Does it have a given H as its subgraph?
- Is this graph planar?
- Is it bipartite?
- Is it k-colorable?
- Does it have good expansion?
- Does it have good clustering?

Testing of Graph Properties

In general – requires at least linear time (often NP-hard)

With a relaxation:
Sublinear-time (or even constant-time) possible



- Does this graph have a clique of size 11?
- Does it have a given H as its subgraph?
- Is this graph planar?
- Is it bipartite?
- Is it k-colorable?
- Does it have good expansion?
- Does it have good clustering?

Testing of Graph Properties

- How can we relax the problem of satisfying a property?

- Optimization problem:
  - For a given graph G, what is its maximum clique size?
    known NP-hard problem
  - We can try to approximate the size

- Decision problem:
  - For a given graph G, is G 3-colorable?
    known NP-complete problem
  - How to relax it?

Testing of Graph Properties

- How can we relax the problem of satisfying a property?
- We want to distinguish
  - between a graph satisfying the property, and
  - a graph that is far from the property

Graph Property Testing definition

- Given input G
- If G has the property ⇨ tester passes
- If G is ε-far from any graph that has the property ⇨ tester fails
- error probability < $1/3$

Notion of **ε-far** : DISTANCE to the Property
One needs to change an ε fraction of the input to obtain an object satisfying the property

Typically we think about $\boldsymbol{\varepsilon}$
as on a small constant, say, $\boldsymbol{\varepsilon = 0.01}$

Graph Property Testing definition

- Given input G
- If G has the property ⇨ tester passes
- If G is ε-far from any graph that has the property ⇨ tester fails
- error probability < $1/3$

- This is two-sided error tester
- one-sided error: errs only for G being ε-far

One sided-error tester often can give a certificate that G doesn't have the property

Framework

- 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 ε-fraction of its representation to obtain a graph with property P*

- What does it mean "an ε-fraction of its representation"?

Testing properties of graphs

Input:

- graph property P
- proximity parameter ε
- input graph $G = (V, E)$ (possibly, from some class of graphs)

Output:

- if G satisfies property P then ACCEPT
- if G is ε-far from having property P then REJECT

Testing properties of graphs

0	1	0	0	1
1	0	1	1	1
0	1	0	0	1
0	1	0	0	0
1	1	1	0	0

Input:

- graph property P
- proximity parameter ε
- input graph $G = (V, E)$ represented by adjacency matrix

Output:

- if G satisfies property P then ACCEPT
- if G is ε-far from having property P then REJECT

Testing properties of graphs

0	1	0	0	1
1	0	1	1	1
0	1	0	0	1
0	1	0	0	0
1	1	1	0	0

Input:

- graph property P
- proximity parameter ε
- input graph $G = (V, E)$ represented by adjacency matrix

Output:

- if G satisfies property P then **ACCEPT**
- if G is ε-far from having property P then **REJECT**

G is ε-far from P if one has to modify $\geq \varepsilon |V|^2$ edges of G to obtain a graph satisfying P

Testing properties of graphs

Input:

- graph property P
- proximity parameter ε
- input graph $G = (V, E)$ of maximum degree d

Output:

- if G satisfies property P then ACCEPT
- if G is ε-far from having property P then REJECT

Testing properties of graphs

Input:

- graph property P
- proximity parameter ε
- input graph $G = (V, E)$ of maximum degree d

Output:

- if G satisfies property P then **ACCEPT**
- if G is ε-far from having property P then **REJECT**

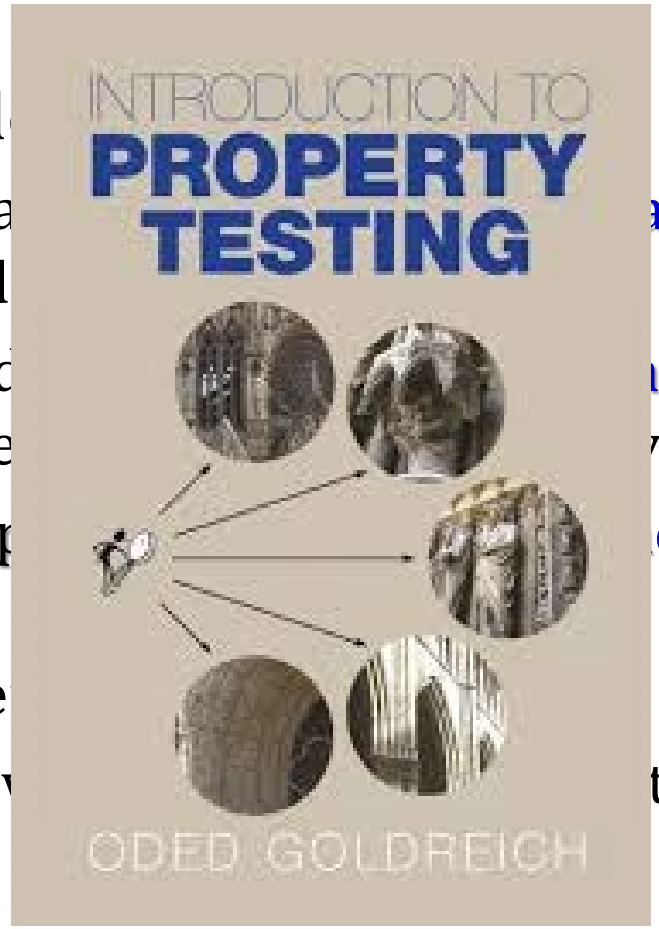
G is ε-far from P if one has to modify $\geq \varepsilon d|V|$ edges of G to obtain a graph satisfying P

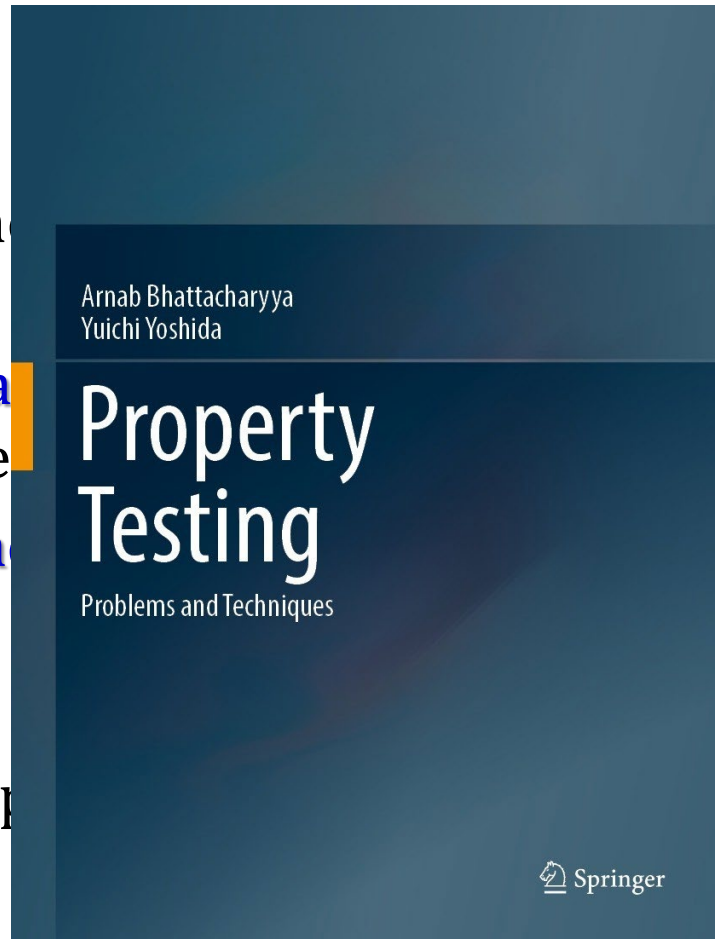
Testing of Graph Properties

- Started with Rubinfeld-Sudan (1996), Goldreich-Goldwasser-Ron (1998), Goldreich-Ron (2002)
- Now we know quite a lot
  - If G is **dense**, given as an oracle to adjacency matrix, then every hereditary property can be tested in constant time
  - If G is **sparse** (bounded degree), given as an oracle to adjacency list, then many properties can be tested in constant time, many can be tested in sublinear time
  - If G is an **arbitrary graph** represented by an oracle to adjacency list, then much less is known
  - If G is directed then very little is known
    - unless there is a trivial reduction to undirected graphs

Testing of Graph Properties

- Started with Rubinfeld-Sudan (1996), Goldreich-Goldwasser-Ron (1998), Goldreich-Ron (2002)
- Now we know quite a l
  - If G is dense, given as a atrix, th property can be tested
  - If G is sparse (bounded cle to a properties can be teste can be
  - If G is an arbitrary grap cle to a less is known
  - If G is directed then ve
    - unless there is a triv ted grap





Framework

- 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 ε-fraction of its representation to obtain a graph with property P*

- What does it mean "an ε-fraction of its representation"?

First model:
Adjacency Matrix

Graph G is ε-far from satisfying property P

If one needs to modify more than ε-fraction of entries in adjacency matrix to obtain a graph satisfying P

Access to G via oracle:
is i connected by edge to j?
($A[i,j] = 1$?)

0	1	0	0	1
1	0	1	1	1
0	1	0	0	1
0	1	0	0	0
1	1	1	0	0

First model:
Adjacency Matrix

Graph G is ε-far from satisfying property P

If one needs to modify more than ε-fraction of entries in adjacency matrix to obtain a graph satisfying P

εn^2 edges have to be added/deleted

Suitable for dense graphs

Adjacency matrix model

0	1	0	0	1
1	0	1	1	1
0	1	0	0	1
0	1	0	0	0
1	1	1	0	0

- Accept every graph that satisfies property P
- Reject every graph that is ε-far from property P
  - ε-far from P: one has to modify at least εn^2 entries of the adjacency matrix to obtain a graph with property P
- Arbitrary answer if the graph doesn't satisfy P nor is ε-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

Adjacency matrix model

0	1	0	0	1
1	0	1	1	1
0	1	0	0	1
0	1	0	0	0
1	1	1	0	0

Very easy example (assume n – large, ε – small constant):

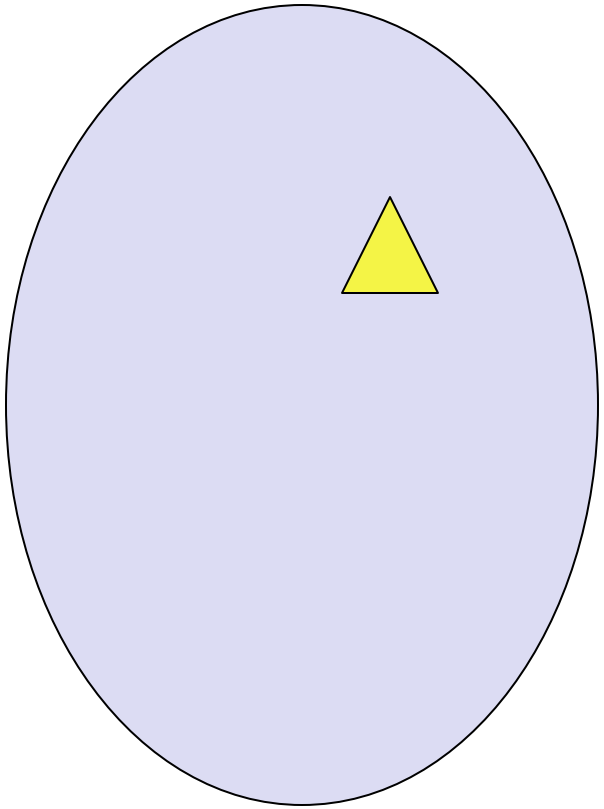
- Test if a graph contains a triangle (cycle of length 3)

Return YES (always)

Highly nontrivial example:

- Test if a graph is triangle-free

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



Adjacency matrix model

0	1	0	0	1
1	0	1	1	1
0	1	0	0	1
0	1	0	0	0
1	1	1	0	0

[Goldreich, Trevisan'03]

Wlog 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 ➔ accept
otherwise ➔ reject

Adjacency matrix model

0	1	0	0	1
1	0	1	1	1
0	1	0	0	1
0	1	0	0	0
1	1	1	0	0

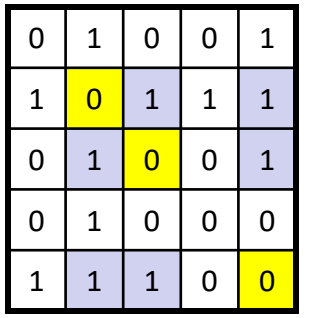
[Goldreich, Trevisan'03]

Wlog 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 ➔ accept
otherwise ➔ reject

Adjacency matrix model



[Goldreich, Trevisan'03]

Wlog 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 ➔ accept
otherwise ➔ reject

Adjacency matrix model

- There are very fast property testers
- They're very simple
- Property tester for bipartiteness:

> •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: $|U| = \tilde{O}(1/\varepsilon) \Rightarrow$ complexity $\tilde{O}(1/\varepsilon^2)$

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
    - bipartiteness
    - planarity
    - being perfect
    - being chordal
    - having no induced subgraph H
    - …

Main Lemma

Main Lemma:

If G is ε-far from satisfying a hereditary property P, then with high probability 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

General result

- 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

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)

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)

$$\text{Tower}(\text{Tower}(\text{Tower}(1/\varepsilon)))$$

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

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

  $O(1/\varepsilon), O(1/\varepsilon^2), O(1/2^{1/\varepsilon})$

Adjacency matrix model

0	1	0	0	1
1	0	1	1	1
0	1	0	0	1
0	1	0	0	0
1	1	1	0	0

Very easy example:

- Test if a graph contains a triangle (cycle of length 3)

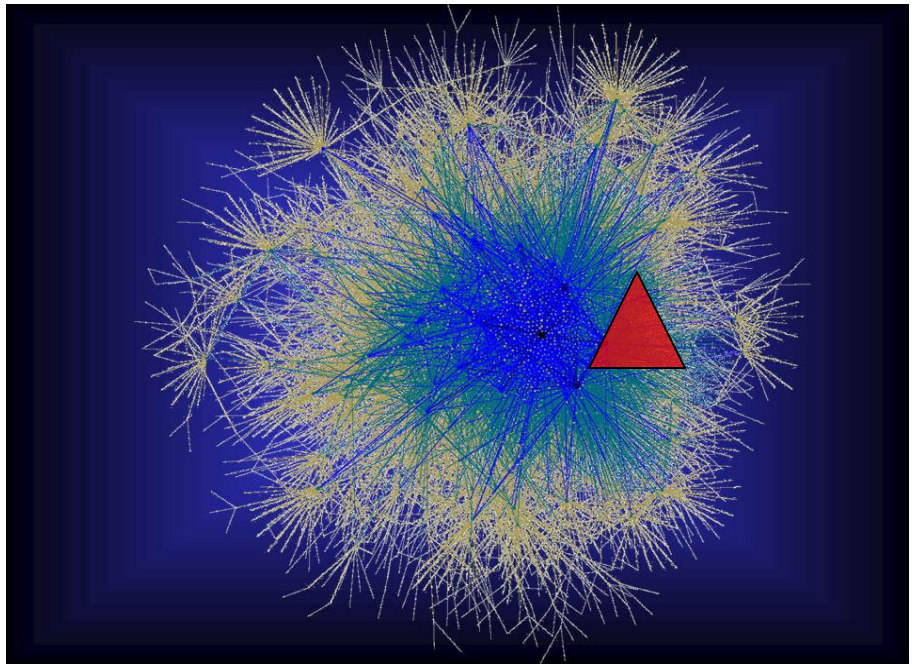
Return YES (always)

Highly nontrivial example:

- Test if a graph is triangle-free

- Can be done in $f(\varepsilon) = O_\varepsilon(1)$ time

- Currently best bound for $f(\varepsilon)$ is Tower(O(log(1/ε)))



Adjacency matrix model

- There are very fast property testers
- They're very simple
- Property tester for bipartiteness:

> •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: $|U| = \tilde{O}(1/\varepsilon) \Rightarrow$ complexity $\tilde{O}(1/\varepsilon^2)$

Adjacency matrix model

Great still-open question: what is the complexity of testing bipartiteness?
We know it's $\widetilde{\Omega}(\varepsilon^{-3/2})$ and $\tilde{O}(\varepsilon^{-2})$

- Th
- They're very simple
- Property tester for bipartiteness:

> •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: $|U| = \tilde{O}(1/\varepsilon) \Rightarrow$ complexity $\tilde{O}(1/\varepsilon^2)$

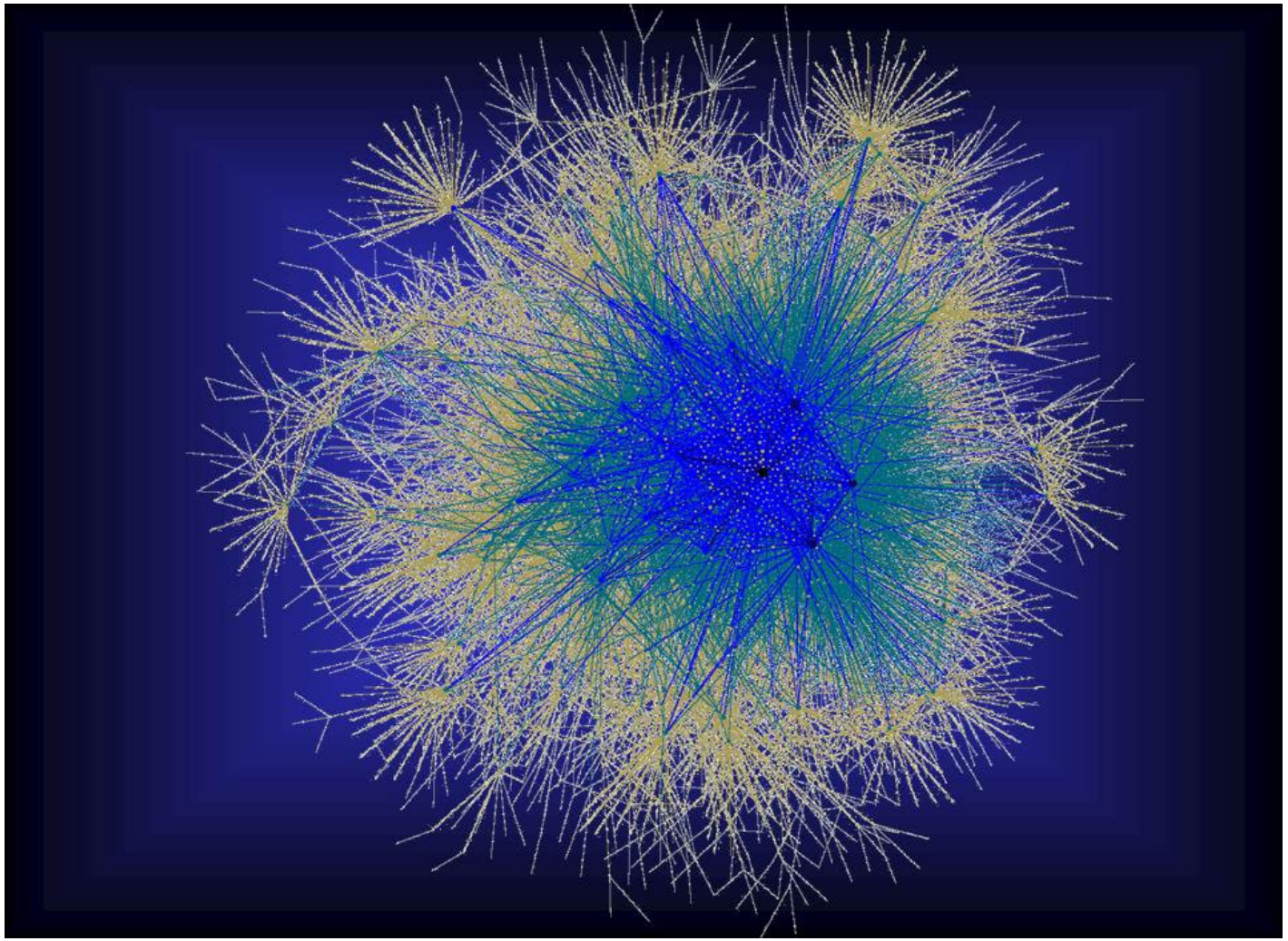
Problems of adjacency matrix model

- Even if many properties are testable in "constant-time", dependency on $1/\varepsilon$ if often very high

- Being ε-far from property requires distance εn^2 from any graph satisfying the property ⇨ distance is big

  – We could reduce the distance by using small ε, but then the dependency on ε would make the complexity very high

Other model ?

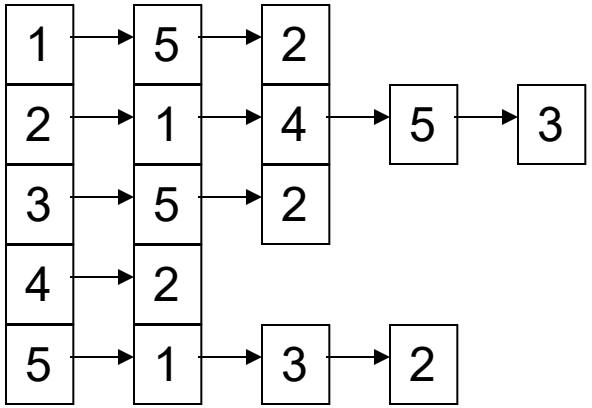


Graph access model

Graph G is ε-far from satisfying property P

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

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



Access to G via oracle:

Return the ith neighbor of v

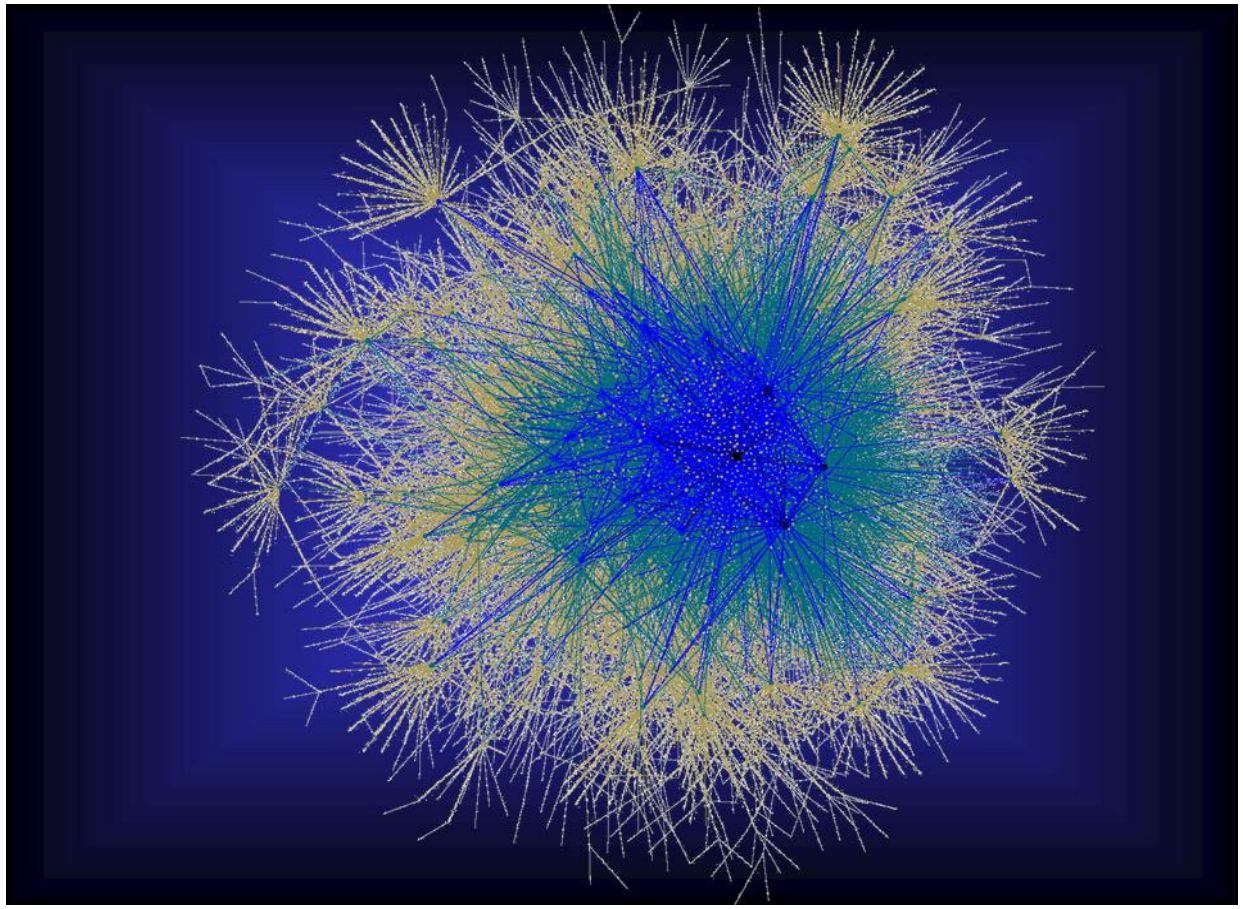
First: Bounded-degree model

- We consider bounded-degree model
  - graph has maximum degree d [constant]
  - ε-far means $\geq \varepsilon dn$ edges to be deleted/added

- 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

Testing connectivity



Bounded-degree adjacency list model

Testing connectivity

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

➔ G has at least εdn connected components

- not enough…we need many small connected components

Bounded-degree adjacency list model

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

G has $\geq \varepsilon dn/2$ connected components of size $\leq 2/\varepsilon d$

Repeat $O(\varepsilon^{-1}d)$ times:
- choose a random vertex v
- run BFS from v until either $1 + 2/\varepsilon d$ vertices have been visited or the entire connected component has been visited
- if v is contained in a connected component of size $\leq 2/\varepsilon d$ then reject

accept

Testing connectivity can be done in $O(\varepsilon^{-2}d)$ time

Bounded-degree adjacency list model

Testing connectivity was easy ...

Similarly easy: testing H-freeness (e.g., triangle-freeness)

- G is ε-far from triangle-free ⇨

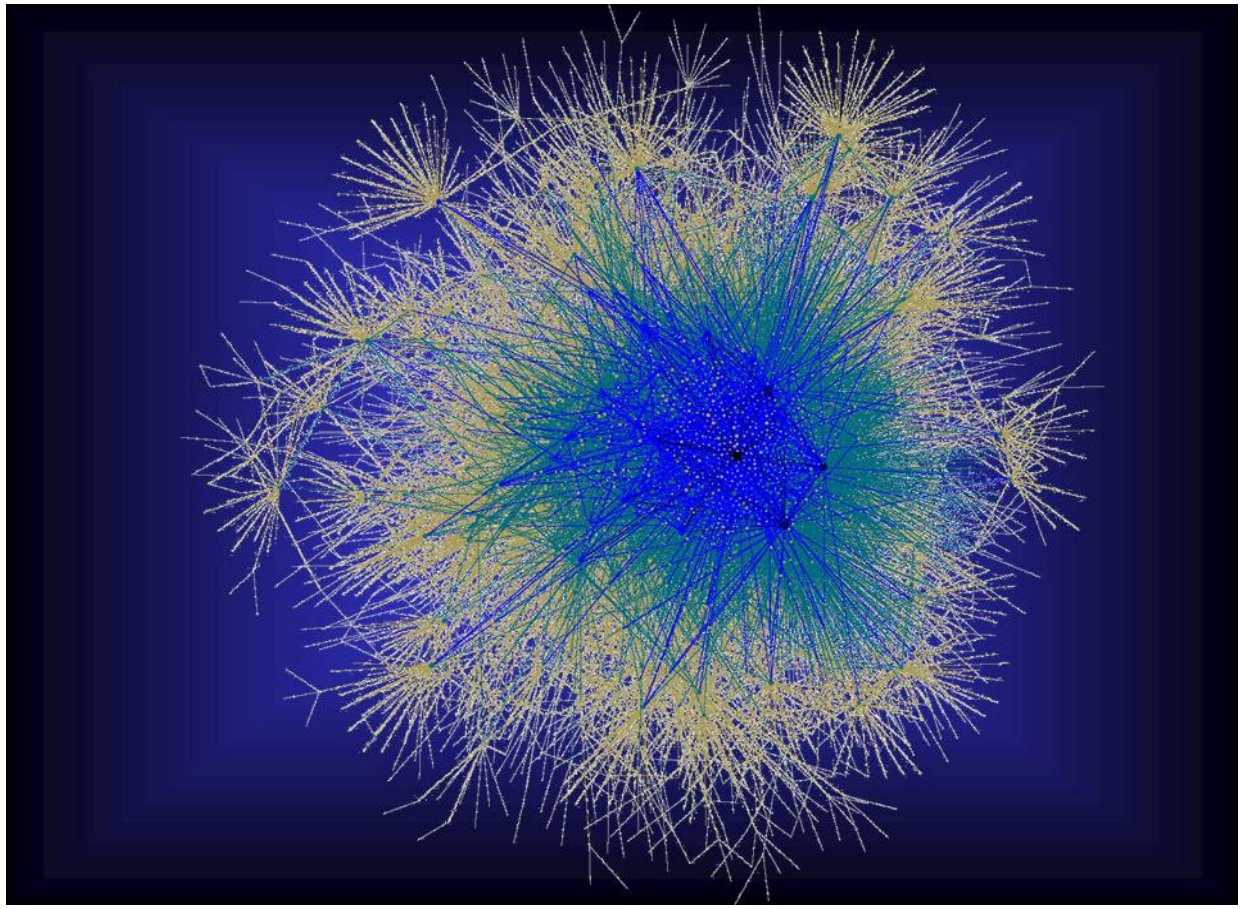
  G has $\Omega(n/\varepsilon)$ disjoint triangles ⇨

  random sampling of $O(1/\varepsilon)$ nodes will detect a triangle

What properties can be tested in constant-time?
We want a characterization!

Bounded-degree adjacency list model

- Testing bipartiteness



Bounded-degree adjacency list model

- Testing bipartiteness
  - Can be done in $\tilde{O}(\sqrt{n}/\varepsilon^{O(1)})$ time (Goldreich & Ron)

Algorithm:

- Select $O(1/\varepsilon)$ starting vertices
- For each vertex run $\text{poly}(\varepsilon^{-1}\log n)\sqrt{n}$ 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 ε-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 bipartiteness
  - Can be done in $\tilde{O}(\sqrt{n}/\varepsilon^{O(1)})$ time (Goldreich & Ron)

Algorithm:

- Select $O(1/\varepsilon)$ starting vertices
- For each vertex run $\text{poly}(\varepsilon^{-1}\log n)\sqrt{n}$ 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: elaborate

- Relatively easy for rapidly mixing case
- For general case: no rapid mixing ⇒ small cut
  use small cut to decompose the graph and the problem

Bounded-degree adjacency list model

- Testing bipartiteness
  - Can be done in $\widetilde{\Theta}(\sqrt{n}/\varepsilon^{O(1)})$ 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

- a Hamiltonian cycle H + a perfect matching M
- a 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 bipartiteness
  - Can be done in $\widetilde{\Theta}(\sqrt{n}/\varepsilon^{O(1)})$ time (Goldreich & Ron)
  - Cannot be done faster (Goldreich & Ron)

So: no constant-time algorithms

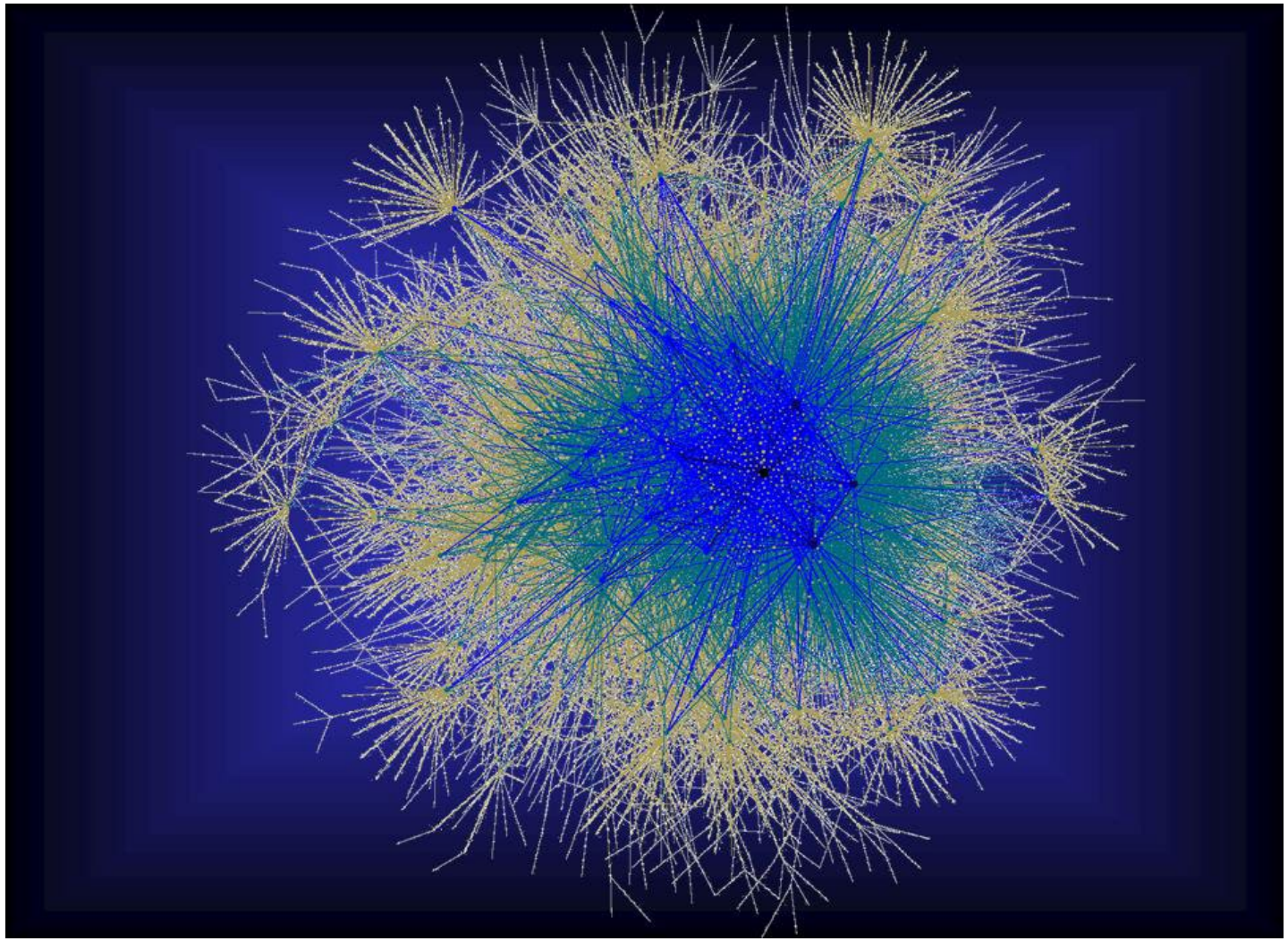
Bounded-degree adjacency list model

- Testing 3-colorability (or k-colorability, $k \geq 3$)

… requires checking (almost) all vertices and edges!

[Bogdanov, Obata, Trevisan'02]

Testing planarity



Testing planarity

Bounded-degree expanders with $\omega(1)$ girth

- There are graphs G such that
  - **any** connected subgraph of G of constant size is planar
  - G is ε-far from planar

- This should mean that we cannot do anything in constant-time …

Testing planarity

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

(with two-sided error) [Benjamini, Schramm, Shapira'08]

For each subgraph of constant size,

- check the number of its occurrences in G
- No all frequencies are possible in planar graphs!
- Some subgraphs cannot appear too often
  wrt some other subgraphs.

Testing planarity

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

(with two-sided error) [Benjamini, Schramm, Shapira'08]

- Runtime: $2^{2^{2^{poly(1/\varepsilon)}}}$
- Hassidim-Kelner-Nguyen-Onak'09 improved the runtime to $2^{\text{poly}(1/\varepsilon)}$
  - with somewhat simpler analysis and simpler algorithm

If G is ε-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

(with two-sided error) [Benjamini, Schramm, Shapira'08]

- Runtime: $2^{2^{2^{poly(1/\varepsilon)}}}$
- Hassidim-Kelner-Nguyen-Onak'09 improved the runtime to $2^{\text{poly}(1/\varepsilon)}$
  - with somewhat simpler analysis and simpler algorithm
- Levi and Ron'13 improved the runtime to $2^{O(\log^2(1/\varepsilon))}$
- Kumar-Seshadhri-Stolman'19 improved it further to $\text{poly}(d \cdot \varepsilon^{-1})$

Testing planarity

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

(with two-sided error) [Benjamini, Schramm, Shapira'08]

- Runt
- Hass
  - wi
- Levi

Consequences (of some follow-up papers):

- poly$(d \cdot \varepsilon^{-1})$-time tester for any property of minor-closed families (e.g., bipartite planar graphs)
- poly$(d \cdot \varepsilon^{-1})$-time algorithms for additive εn-approximations for problems such as maximum matching, minimum vertex cover, maximum independent set, and minimum dominating set for these graph families

- Kumar-Seshadhri-Stolman'19 improved it further to poly$(d \cdot \varepsilon^{-1})$

Testing planarity

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

(with two-sided error)

[Benjamini, Schramm, Shapira'08]
[Hassidim, Kelner, Nguyen, Onak'09]
[Levi, Ron'13]
[Kumar-Seshadhri-Stolman'19]

- Runtime: $\text{poly}(d \cdot \varepsilon^{-1})$ (constant for $d, \varepsilon = O(1)$)
- The result is with two-sided error:
  - can accept non-planar graphs & can reject planar graphs
- There is no $o(\sqrt{n})$-time one-sided-error tester for planarity
- Kumar-Seshadhri-Stolman'18: $O(n^{1/2+o(1)})$-time one-sided-error planarity tester

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

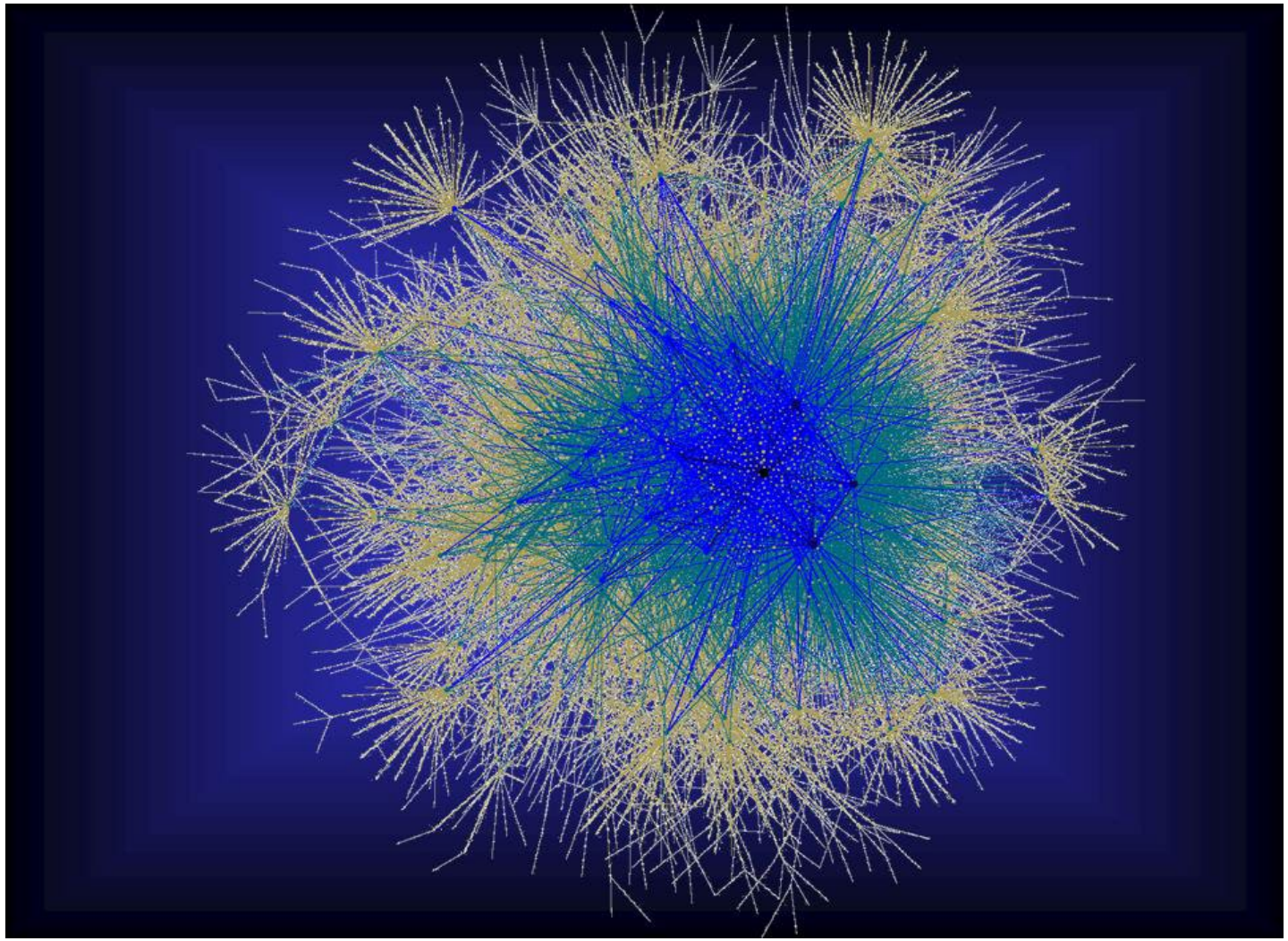
Testing expansion

- In the adjacency list model, rapidly mixing properties play key role:
  - If G doesn't "mix" fast then … testing is fast

- Planar graphs don't mix fast (have large cuts)
  - Testing properties in planar graphs might be easy
- Expanders mix fast:
  - Testing properties might be hard

Expander $G = (V, E)$:
each vertex set U $\subseteq V$ has large neighborhood (proportion to $|U|$)

Testing expansion



Testing expansion

- For graphs of bounded degree, we can distinguish expanders from graphs that are "far" even from poor expanders in $O^*(\sqrt{n})$ time

[C, Sohler '07, Kale, Seshadhri'07, Nachmias, Shapira'08]

- $\Omega(\sqrt{n})$ time is needed

[Goldreich, Ron'02]

Testing expansion

Choose $O(1/\varepsilon)$ nodes at random
For each chosen node run $O(\sqrt{n})$ random walks of length $O(\log n)$
    Count the number of collisions at the end-nodes
    If the number of collisions is too large then **Reject**
Accept

Idea:

- If G is an expander then end-nodes are random nodes

    ⇨ we can estimate number of collisions well

- If G is far from expander then we will have many more collisions (requires non-trivial arguments)

Testing expansion and clustering

In a similar time we can:

- Correct an almost expander-graph to become a good expander, even in distributed way
- Determine graph clustering into expanders
- …

Testing in planar graphs

- All previous results assumed the input graph is arbitrary

Testing in planar graphs is easier!

- Testing bipartiteness in planar graphs of bounded degree can be done in constant time

[C, Sohler, Shapira'09]

Pick random sample of $O^*(d/\varepsilon)$ vertices
For each vertex explore its neighborhood (of size $(d/\varepsilon)^{O(1)}$)
If the input graph is ε-far from bipartite:
the induced subgraph should NOT be bipartite!

Complexity/runtime
$\boldsymbol{(d/\varepsilon)^{O(d/\varepsilon)^{O(1)}}}$

Testing in hyperfinite graphs

- One can go beyond planar graphs:
  - It's enough to have **some separator properties**

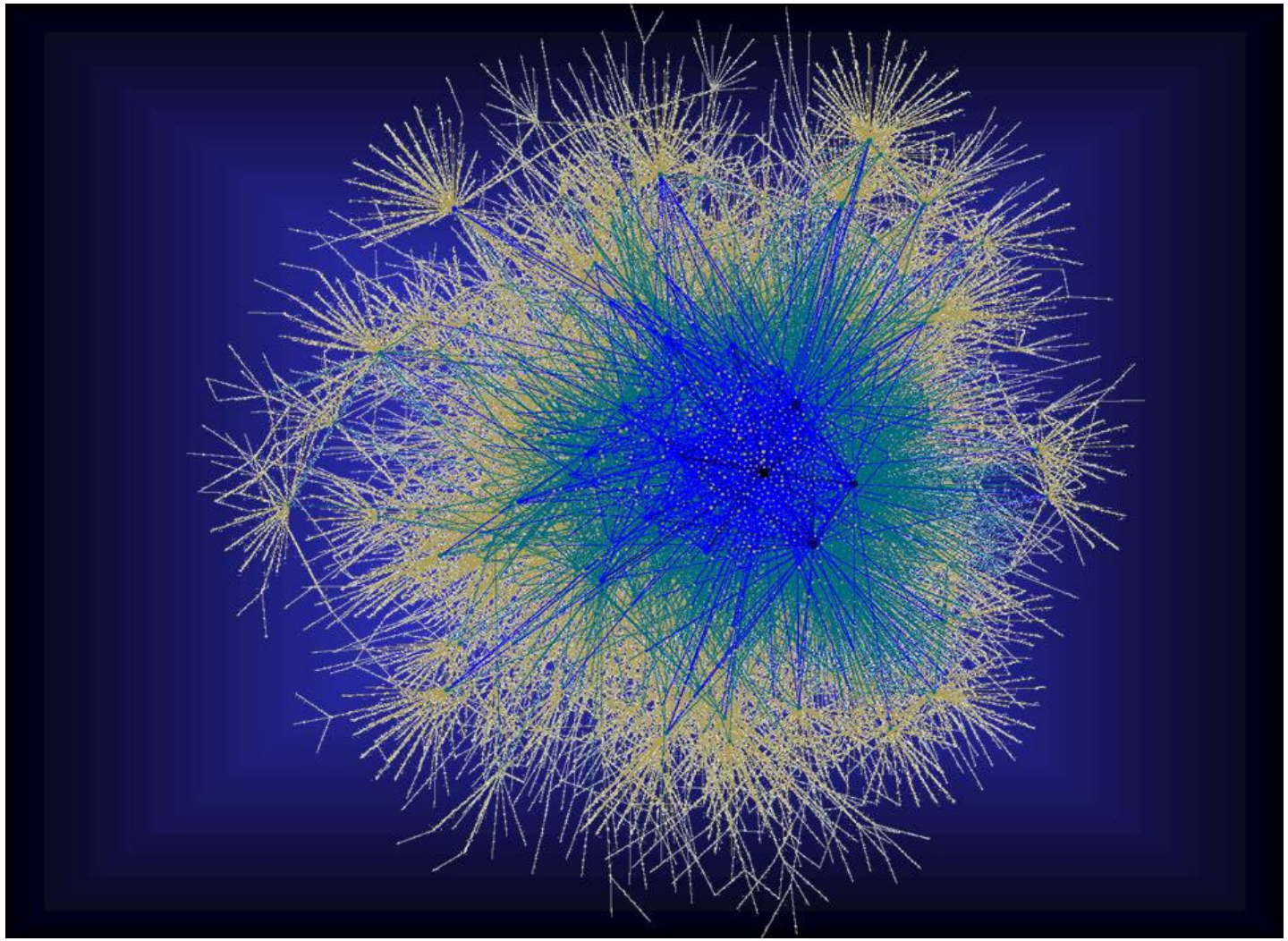
Testing in hyperfinite graphs

Complete characterization for *non-uniform* algorithms:

Newman & Sohler'2011:

- Testing any property in hyperfinite ("non-expanding") families of graphs of bounded-degree can be done in $O(1)$ time (two-sided-error)

Arbitrary graphs (no bound for max-degree)



Adjacency Lists in arbitrary graphs

Graph G is ε-far from satisfying property P

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

More general model & more challenging model:
graphs of arbitrary max-degree

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

Adjacency Lists in arbitrary graphs

Oracle access:

- Sample a random vertex
- Sample a random neighbour of a given vertex

  (could be replaced by two queries: degree query + i^{th} neighbour)

Graph $G = (V, E)$ is ε–far from property P if one has to modify $\geq \varepsilon|E|$ edges in G to obtain a graph satisfying P

These techniques don't work for arbitrary-degree graphs

Previous techniques don't work for arbitrary-degree graphs:

Testing planarity (or H-freeness) in arbitrary degree graphs requires $\Omega(\sqrt{n})$ time, even in two-sided error model

Two instances:

- empty graph on n nodes
- clique on $\sqrt{n}$ nodes + isolated $n - \sqrt{n}$ nodes

Testing in arbitrary graphs

- Testing neighborhood may cost even $O(n)$ time!

➔ Graph exploration is expensive

Testing in arbitrary graphs

- Almost nothing is known

- Triangle-freeness
  - Alon et al'2008
- Bipartiteness
  - Ben-Eliezer et al'2008
  - If G has average degree d then $\Omega(\min\{\sqrt{n}, n/d\})$ queries are needed

Testing in arbitrary planar graphs

- C, Monemizadeh, Onak, Sohler'11

- Testing bipartiteness in planar graphs can be done in constant time

- Challenge:

  how to explore neighbourhood of a node quickly?

> ➢ Run many short random walks
>
> ➢ For a planar graph that is ε-far from bipartite, prove that one of the random walks will find an odd-length cycle

Testing in arbitrary planar graphs

- C, Monemizadeh, Onak, Sohler'11

Challenge: analyze random walks of constant length

first step: reduce to testing C_k-freeness for odd k, all $k = O(1/\varepsilon)$

- Run many short random walks
- For a planar graph that is ε-far from bipartite, prove that one of the random walks will find an odd-length cycle

Testing in arbitrary planar graphs: it's all about H-freeness

- Broader class of graphs than planar
  - Graphs defined by arbitrary fixed forbidden minors

C. & Sohler'19:

- Testing $\boldsymbol{H}$-freeness in general planar graphs can be done in constant time with one-sided error (for arbitrary finite H)
- P is testable in constant-time with one-sided error in planar graphs iff P "can be reduced" to test H-freeness

Characterization of properties testable in constant-time (for planar graphs)

Testing in arbitrary planar graphs

C. & Sohler'19:

- Testing $\boldsymbol{H}$-freeness in general planar graphs can be done in constant time with one-sided error (for arbitrary finite H)
- P is testable in constant-time with one-sided error in planar graphs iff P "can be reduced" to test H-freeness

Characterization of properties testable in constant-time (for planar graphs)

Esperet & Norin'22: for any proper minor-closed class $\mathcal{G}$, any monotone property is testable for graphs from $\mathcal{G}$

(e.g., for any k and t, k-colorability of K_t-minor free graphs is testable; monotone properties of graphs from minor-closed classes that are closed under disjoint union can be tested in constant time)

Testing of Digraph Properties

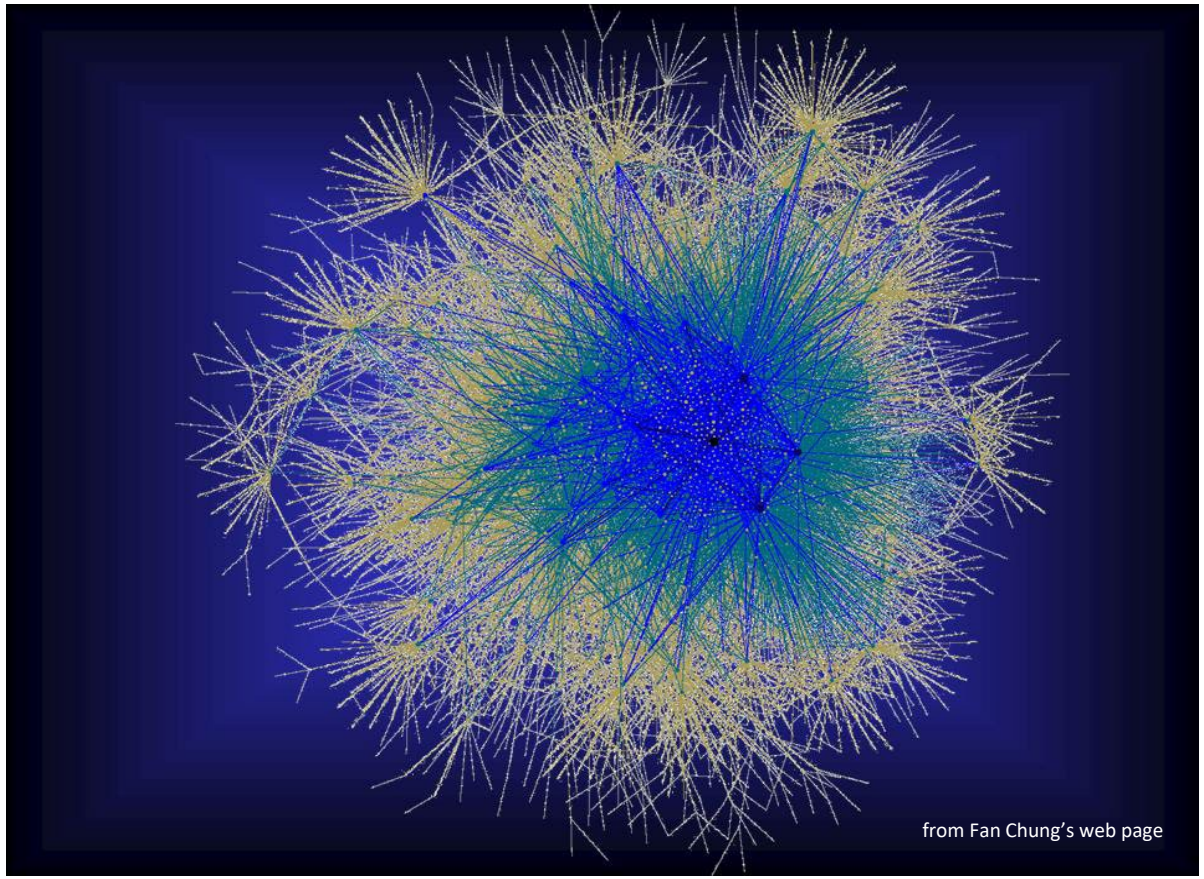
Models introduced by Bender-Ron (2002):

- Digraphs with bounded maximum in- and out-degrees
- Oracle with access to adjacency list
- Two main models:
  - Bidirectional: outgoing and incoming edges
    - shares properties of undirected graphs;
    - not suitable in many scenarios/applications
  - One-directional: access to outgoing edges only
    - major difference wrt undirected graphs
    - more natural in many scenarios/applications

Sometimes very fast

More challenging

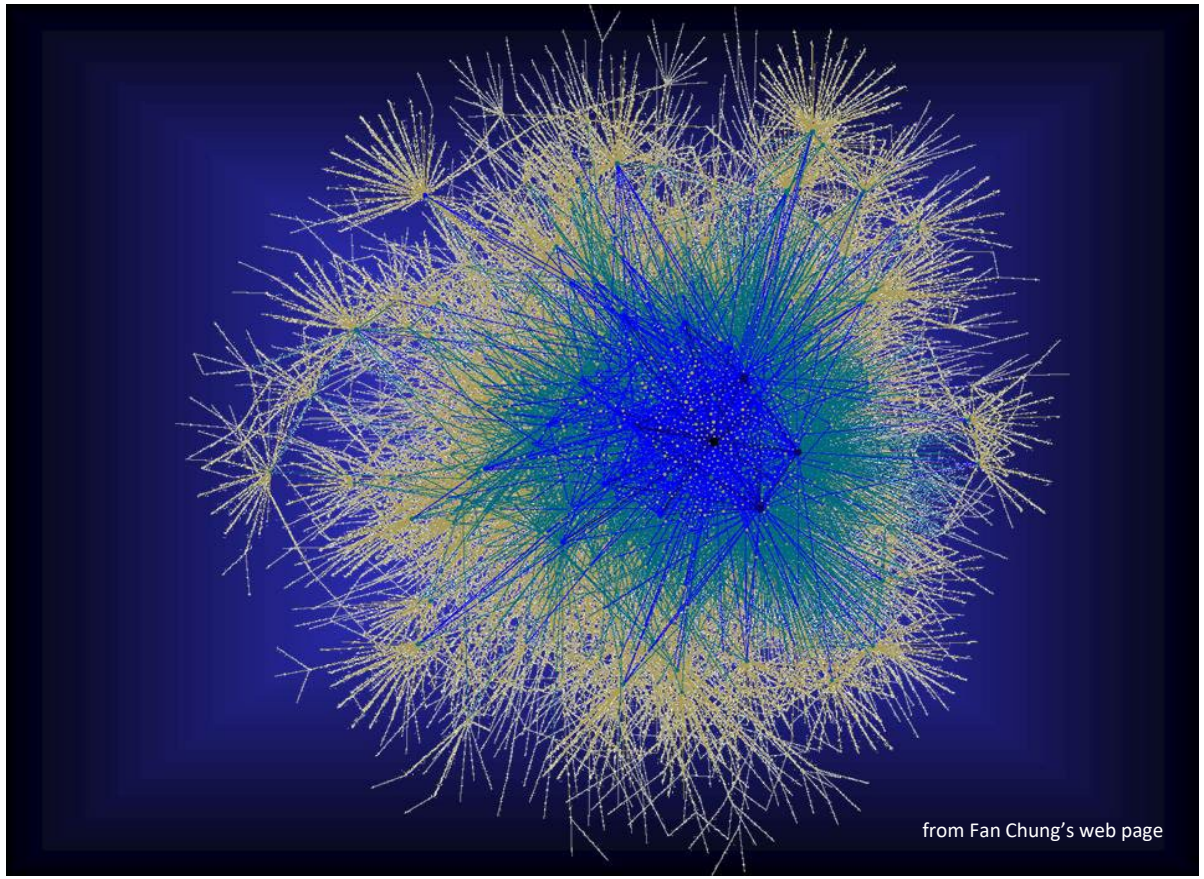
Big networks



- Is it weakly connected?
  (or close to it)
- Is it planar?
  (or close to it)

If we have access to both directional edges then this reduces to a problem in undirected graphs (which we understand well)

Big networks



- Is it strongly connected?
  (or close to it)
- Is it acyclic?
  (or close to it)
- Is it C_{33}-free?
  (or close to it)

Highly non-trivial if we have no access to incoming edges
For example: we cannot easily check if a node has in-degree 0

Relation between the models
C, Peng, Sohler'16 + Peng, Wang'23

There is a tester for property P with constant query time in bidirectional model

We can test P in one-directional model with sublinear $n^{1-\Omega_{\varepsilon,d}(1)}$ query time (in two-sided error model)

Relation between the models
C, Peng, Sohler'16

There is a tester for property P with constant query time in bidirectional model

We can test P in one-directional model with sublinear $n^{1-\Omega_{\varepsilon,d}(1)}$ query time (in two-sided error model)

Application:
Every hyperfinite (e.g., planar) property can be tested with sublinear complexity in one-directional model

Relation between the models
C, Peng, Sohler'16

There is a tester for property P with constant query time in bidirectional model

We can test P in one-directional model with sublinear $n^{1-\Omega_{\varepsilon,d}(1)}$ query time (in two-sided error model)

This cannot be improved much:

- two-sided error is required (cf. strong connectivity)
- $\Omega(n^{1-1/k})$ "simulation" slowdown is required (cf. 3-star-freeness)

Summary

- Many graph properties can be tested efficiently (in the property testing framework)
  - Sometimes in constant-time
  - More often in sublinear-time

- But our understanding of testing graph properties
  - in arbitrary graphs or in directed graphs is still patchy …

- Tools:
  - Combination of algorithm design, combinatorics, analysis of random walks …

Further topics (not discussed here): Tolerant Testing

- Setting so far: with probability at least $2/3$, determine whether
  - G has property P, or
  - G is ε-far from any graph having property P.
- Feature: freedom to answer if G is close to having P but does not exactly have this property.

- *Tolerant testing*: separate inputs that are ε-far from having property P from those that are ε'-close to having property P, where $\varepsilon' > 0$.
  - We call this a $(\varepsilon', \varepsilon)$-tolerant tester
  - Introduced by Parnas, Ron, Rubinfeld'06
  - Related to distance approximation

Further topics (not discussed here): *Distributed* Property Testing

Some examples, include:

DISC 2016:

- Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev, “Fast distributed algorithms for testing graph properties”
- Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca, “Distributed testing of excluded subgraphs”

DISC 2017

- Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca, “Three notes on distributed property testing”

PODC 2018

- Reut Levi, Moti Medina, Dana Ron, “Property testing of planarity in the CONGEST model”

Local Computation Algorithms (LCAs)

Further topics (not discussed here): Optimization algorithms

- Property testing tools lead to fast optimization algorithms
- Estimating the weight of the minimum spanning tree:
  - Chazelle, Rubinfeld, Trevisan (2001): $(1+\varepsilon)$-approximation algorithm in $O(DW/\varepsilon^3)$ time, D-max-degree; W-max-weight
  - C, Sohler (2004): $(1+\varepsilon)$-approximation algorithm in $O(n \cdot \log^{O(1)} n/\varepsilon^{O(1)})$ time for arbitrary metric graphs

- In a graph of constant degree, in constant time:
  - we can estimate the cost of the **minimum vertex cover** to within factor of 2 and additive error term εn
  - we can estimate the **size** of the **maximum matching** with additive error term εn

THANK YOU!