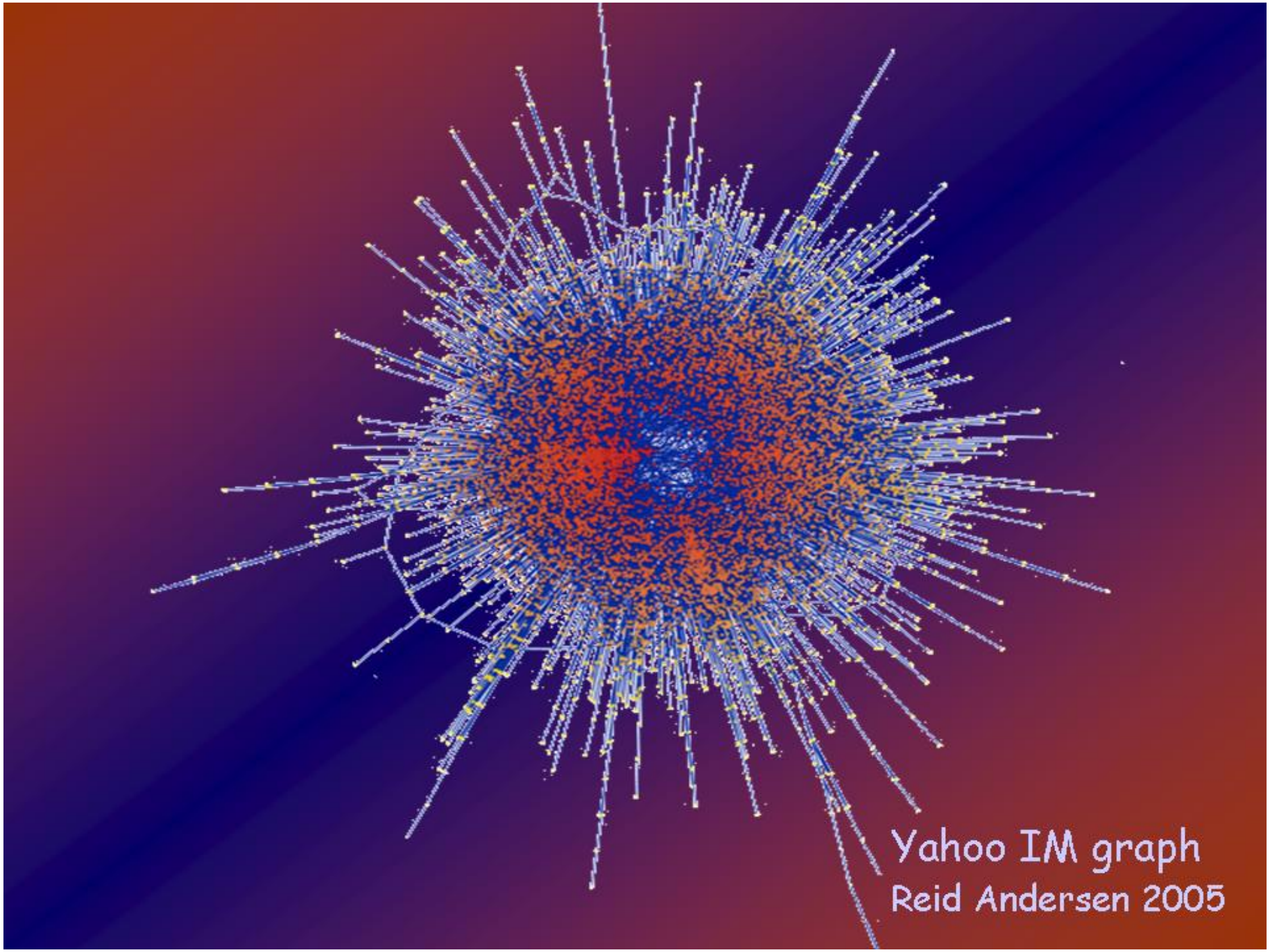
Fast Testing of Graph Properties

Artur Czumaj
DIMAP and Department of Computer Science
University of Warwick

Fast Testing of Graph Properties (in Big Data setting)

- We want to process Big Graphs quickly
  - Detect basic properties
  - Analyze their structure

Fast Testing of Graph Properties
(in Big Data setting)



Fast Testing of Graph Properties (in Big Data setting)

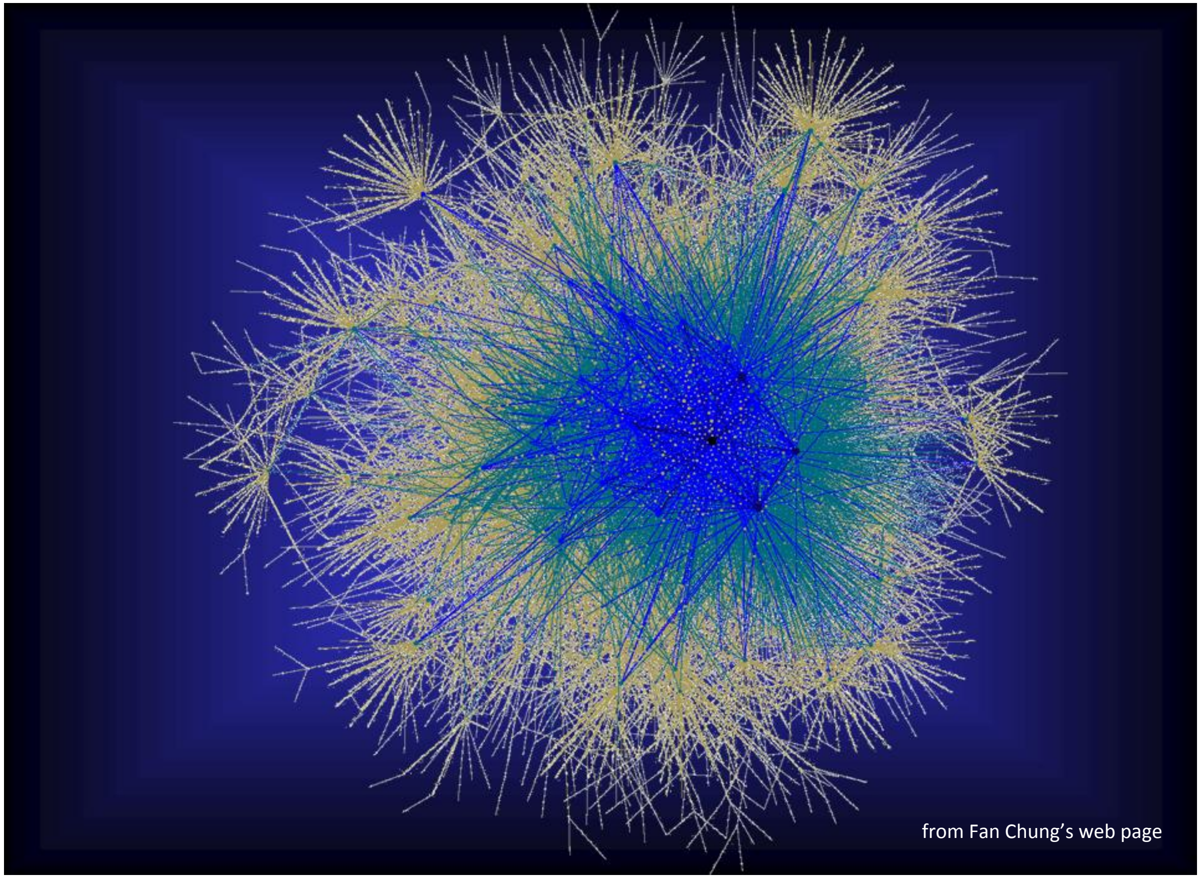
- We want to process Big Graphs quickly
  - Detect basic properties
  - Analyze their structure

- For graphs with millions or billions of nodes, quickly often mean *sublinear* in the size of the graph

Fast Testing of Graph Properties

- How to test basic properties of graphs
  in the framework of property testing

Fast Testing of Graph Properties



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

Framework of property testing

- We cannot quickly give 100% precise answer
- We need to approximate

- Distinguish graphs that have specific property from those that are far from having the property

Property Testing definition

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

Notion of $\boldsymbol{\varepsilon}$-far : DISTANCE to the Property

One needs to change ε fraction of the input to obtain an object satisfying the property

Typically we think about $\boldsymbol{\varepsilon}$ as on a small constant, say, $\boldsymbol{\varepsilon = 0.1}$

Property Testing definition

- Given input G
- If G has the property ⇨ tester passes
- If G is ε-far from any string 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"?

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

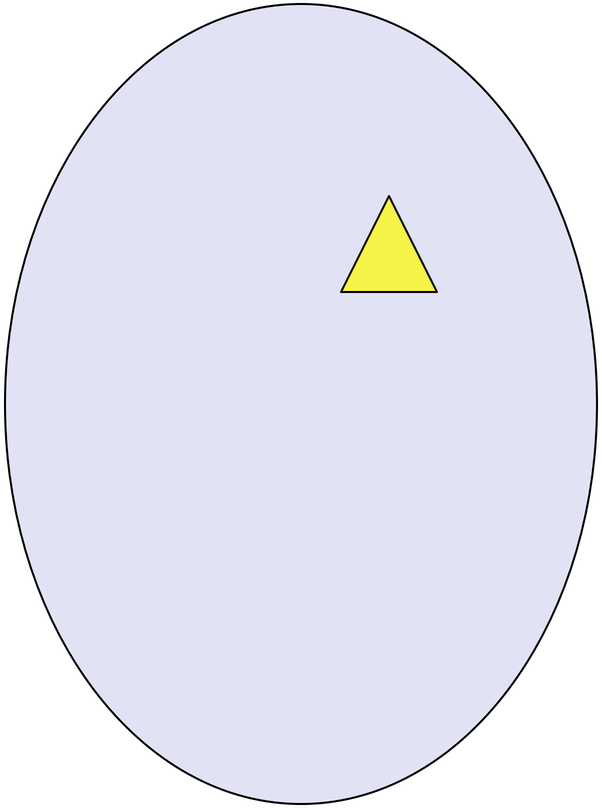
Very easy example:

- 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

- 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: $|U| = poly(1/\varepsilon)$

•Alon, Krivelevich: $|U| = O^*(1/\varepsilon) \Rightarrow$ 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
    - ...

Main Lemma

Main Lemma:

If G is ε-far from satisfying a hereditary property P, then whp random subgraph of size $WP(\varepsilon)$ 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)

$$2^{2^{\cdot^{\cdot^{\cdot^{2^{1/\varepsilon}}}}}}$$

three or more towers of height $O(1/\varepsilon)$

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)

$$\text{Tower}(c) = 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}} \Big\} c \text{ times}$$

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)

$$Tower(Tower(Tower(1/\varepsilon)))$$

For $\varepsilon = 0.5$ *we have* $Tower(Tower(Tower(1/\varepsilon))) = Tower(65536)$

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

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

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)

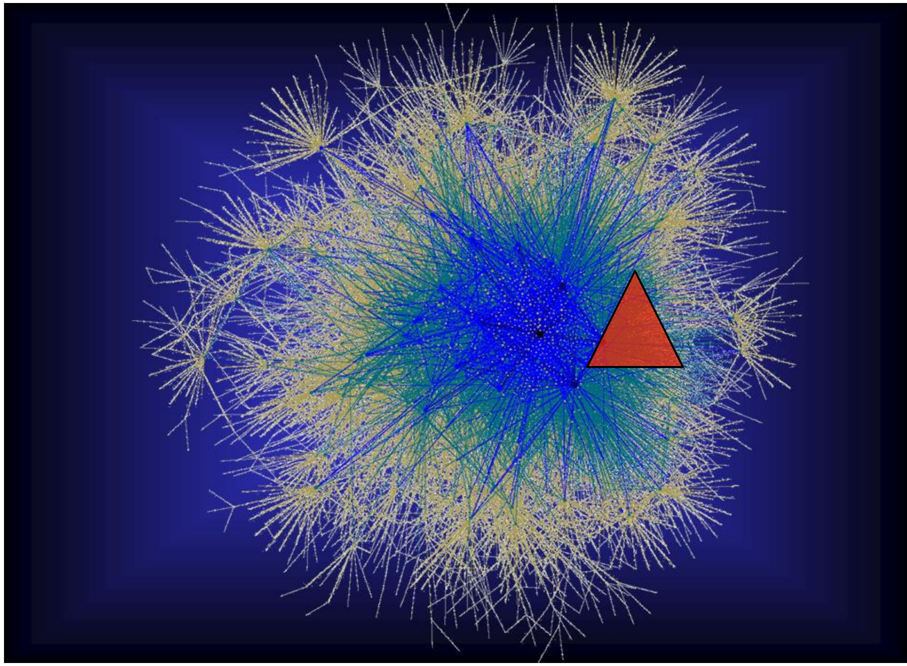
Return YES (always)

Highly nontrivial example:

- Test if a graph is triangle-free

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

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

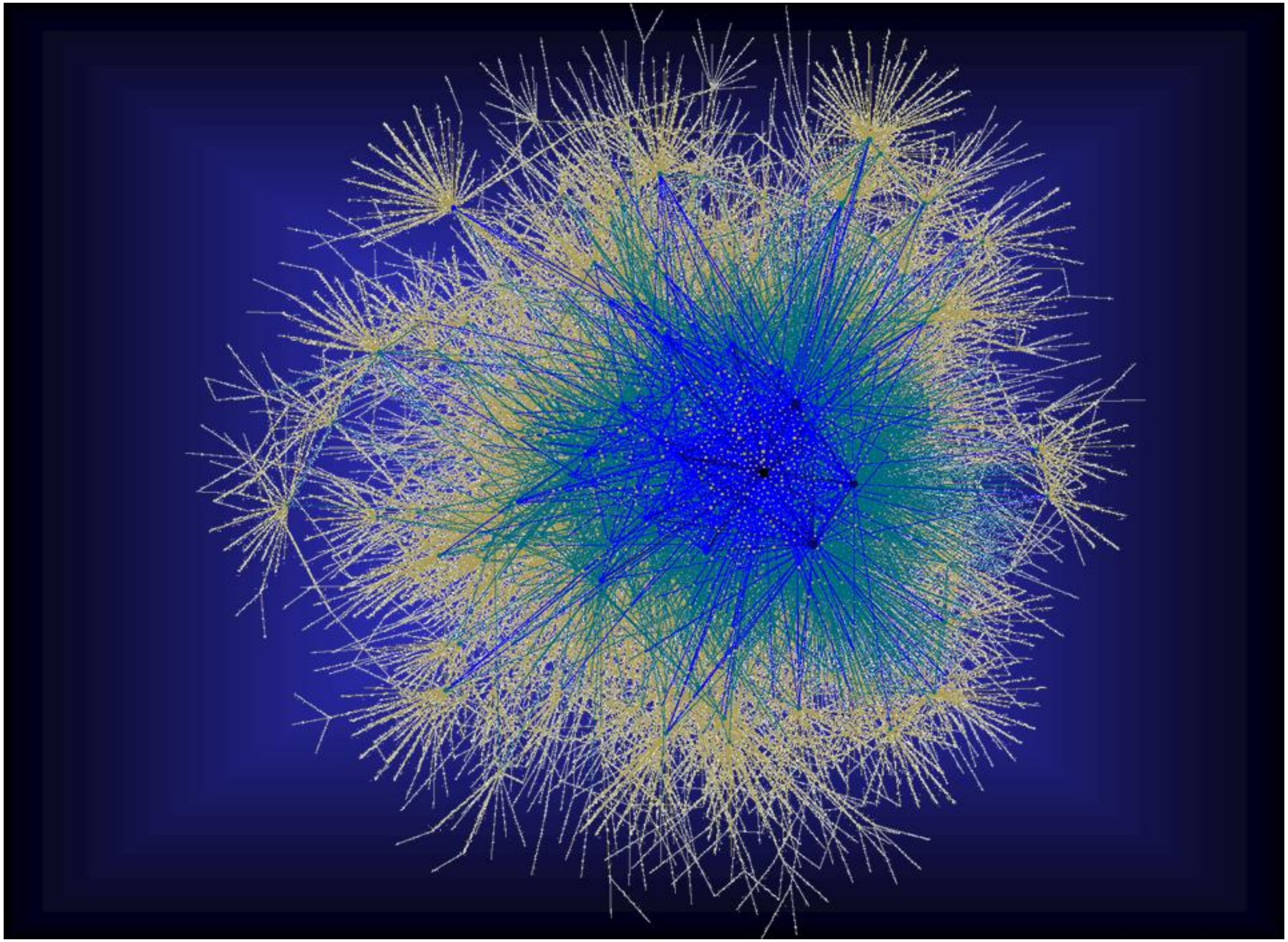


Problems of adjacency matrix model

- 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

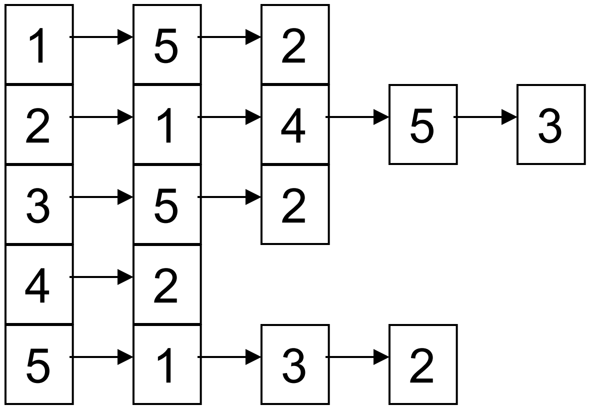
Other model ?



Second model: Adjacency Lists

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



Access to G via oracle:

Return the ith neighbor of v

Second model: Adjacency Lists

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

Suitable for sparse graphs

Second model: Adjacency Lists

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

Main model: graphs with max-degree d

εdn edges have to be added/deleted

Bounded-degree model

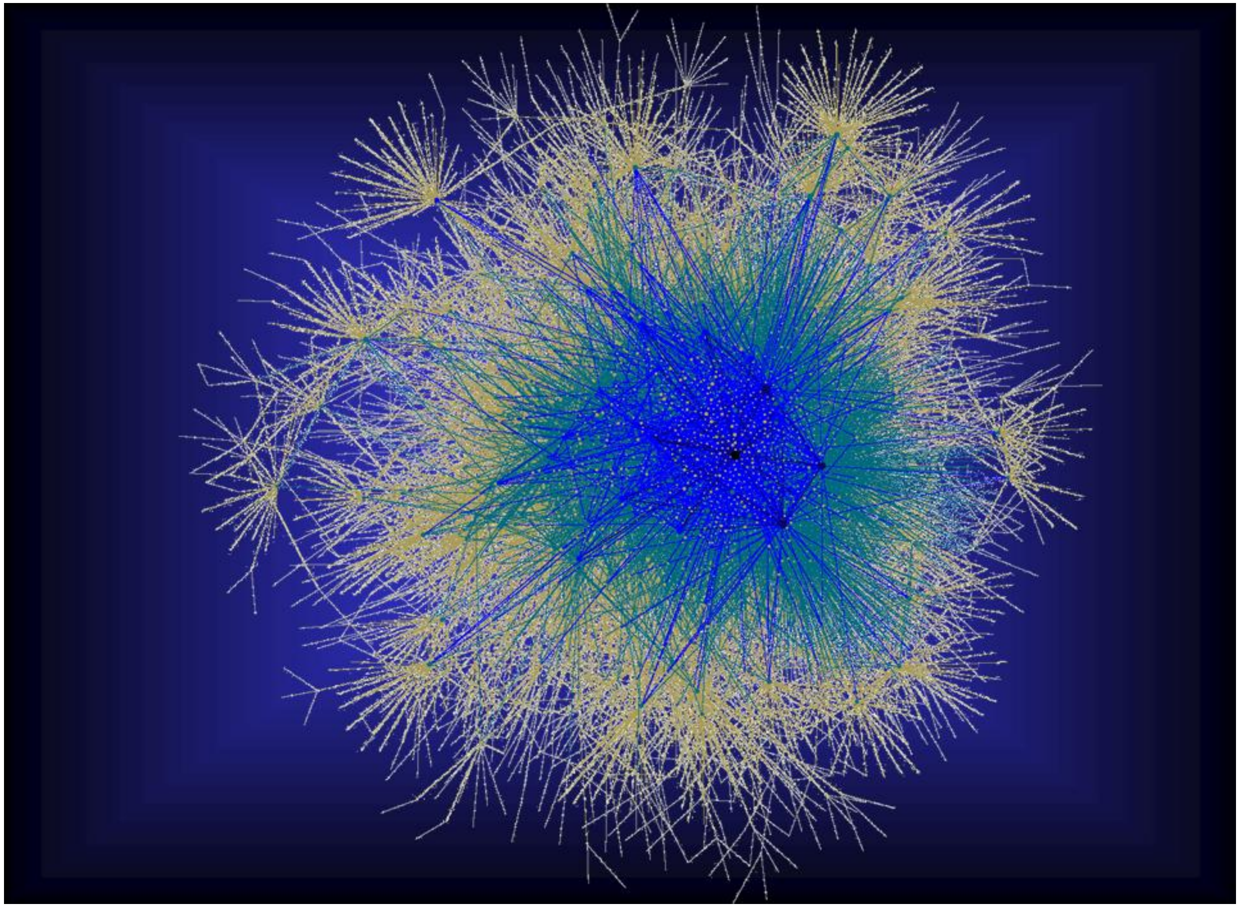
- We consider bounded-degree model
  - graph has maximum degree d [constant]

- 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

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 ...

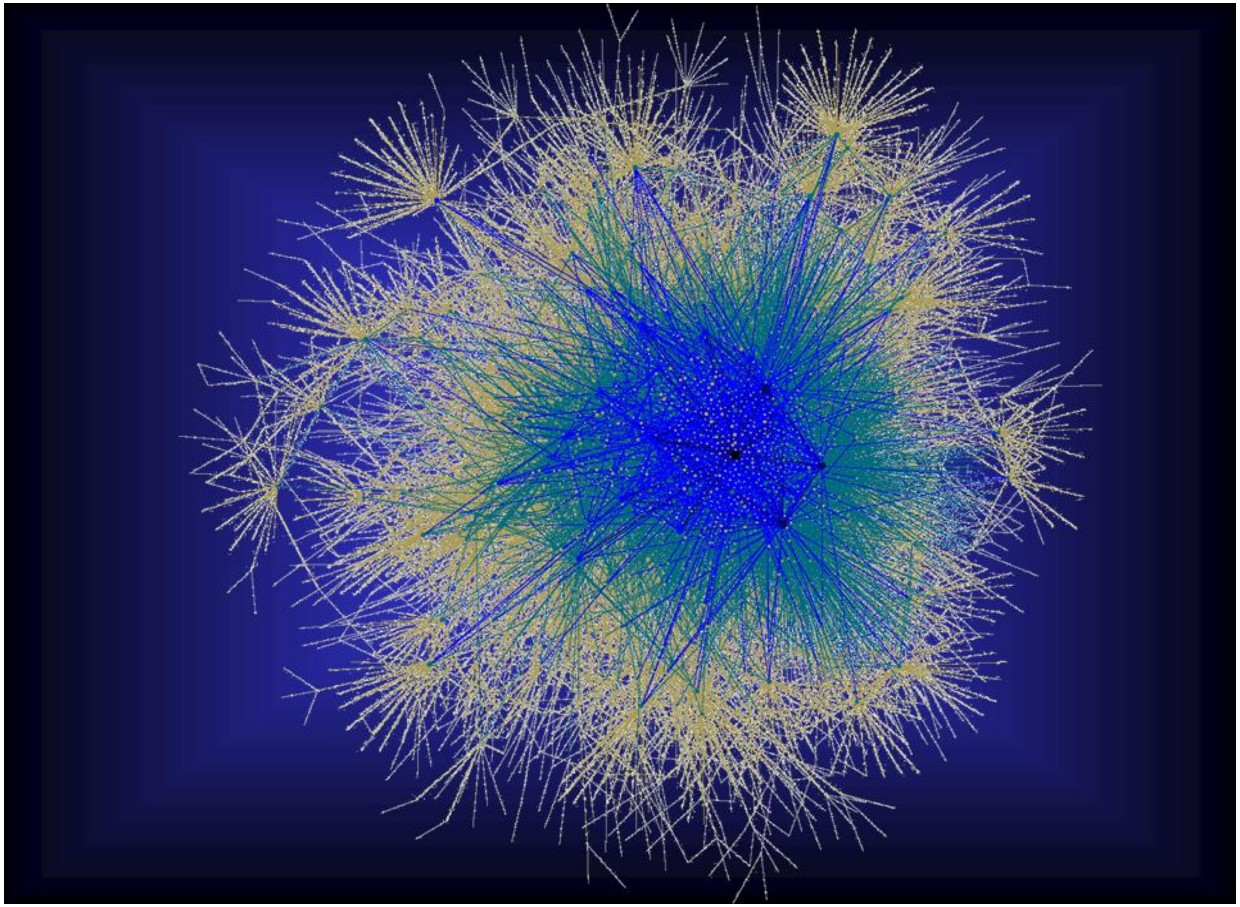
Similarly easy: testing H-freeness (e.g. triangle-freeness)

- 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 $\mathbf{O}^*(\sqrt{\mathbf{n}}/\boldsymbol{\varepsilon}^{\mathbf{O}(\mathbf{1})})$ time (Goldreich & Ron)

Algorithm:

•Select $O(1/\varepsilon)$ starting vertices

•For each vertex run poly$(\varepsilon^{-1}\log n)\sqrt{n}$ random walks of length 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 $O^*(\sqrt{n}/\varepsilon^{O(1)})$ time (Goldreich & Ron)

Algorithm:

•Select $O(1/\varepsilon)$ starting vertices

•For each vertex run poly$(\varepsilon^{-1}\log n)\sqrt{n}$ random walks of length 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 ⇒ small cut

  use small cut to decompose the graph and the problem

Bounded-degree adjacency list model

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

So: no constant-time algorithms

Bounded-degree adjacency list model

- Testing 3-colorability

… requires checking (almost) all vertices and edges!

[Bogdanov, Obata, Trevisan'02]

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 $\Omega(\sqrt{n})$

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)

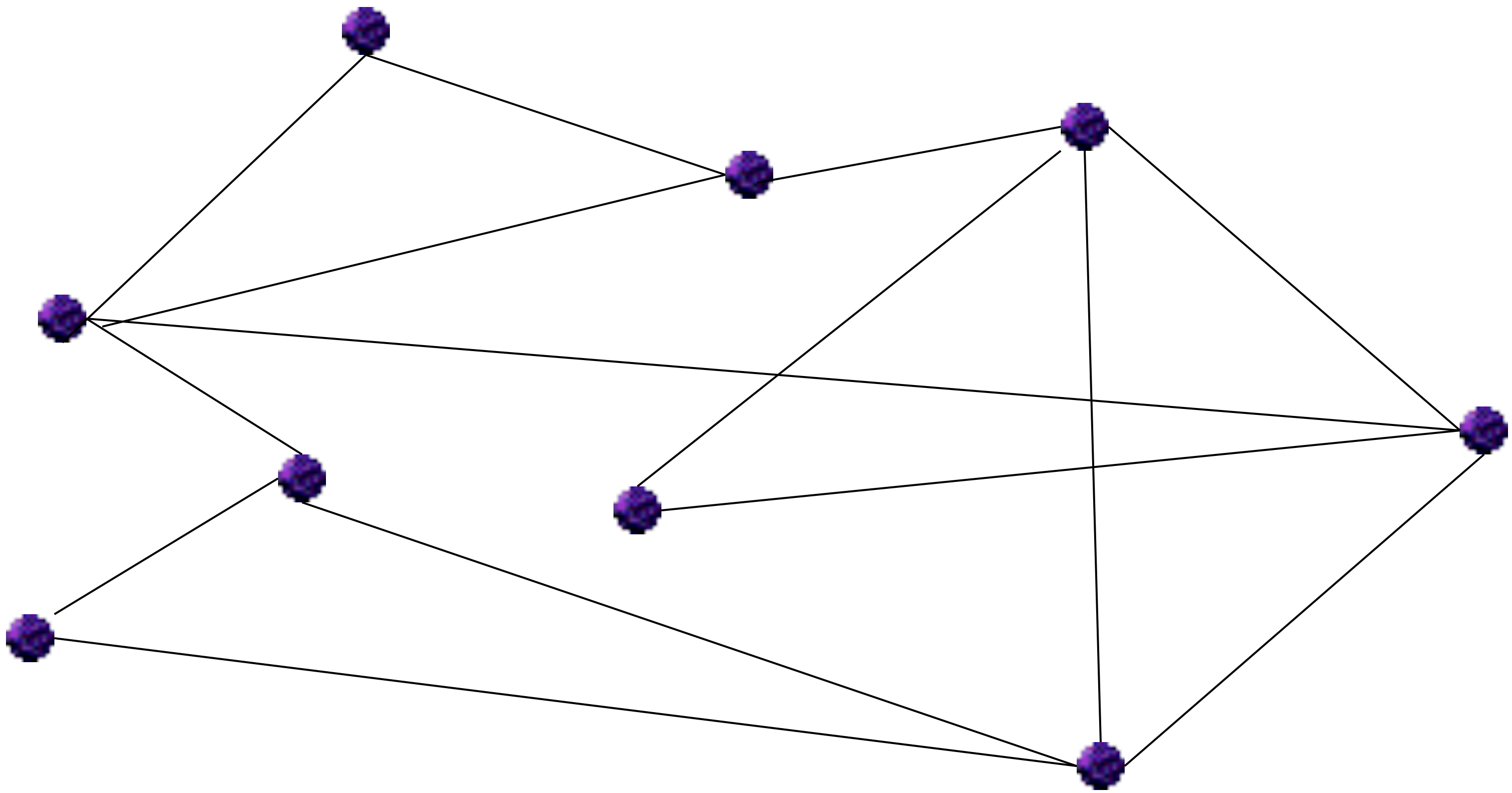
C, Goldreich, Ron, Seshadhri, Sohler, Shapira '12
An upper bound of $O^*(\sqrt{n})$:
reduction to bipartiteness

Testing cycle-freeness in bounded-degree graphs one-sided error

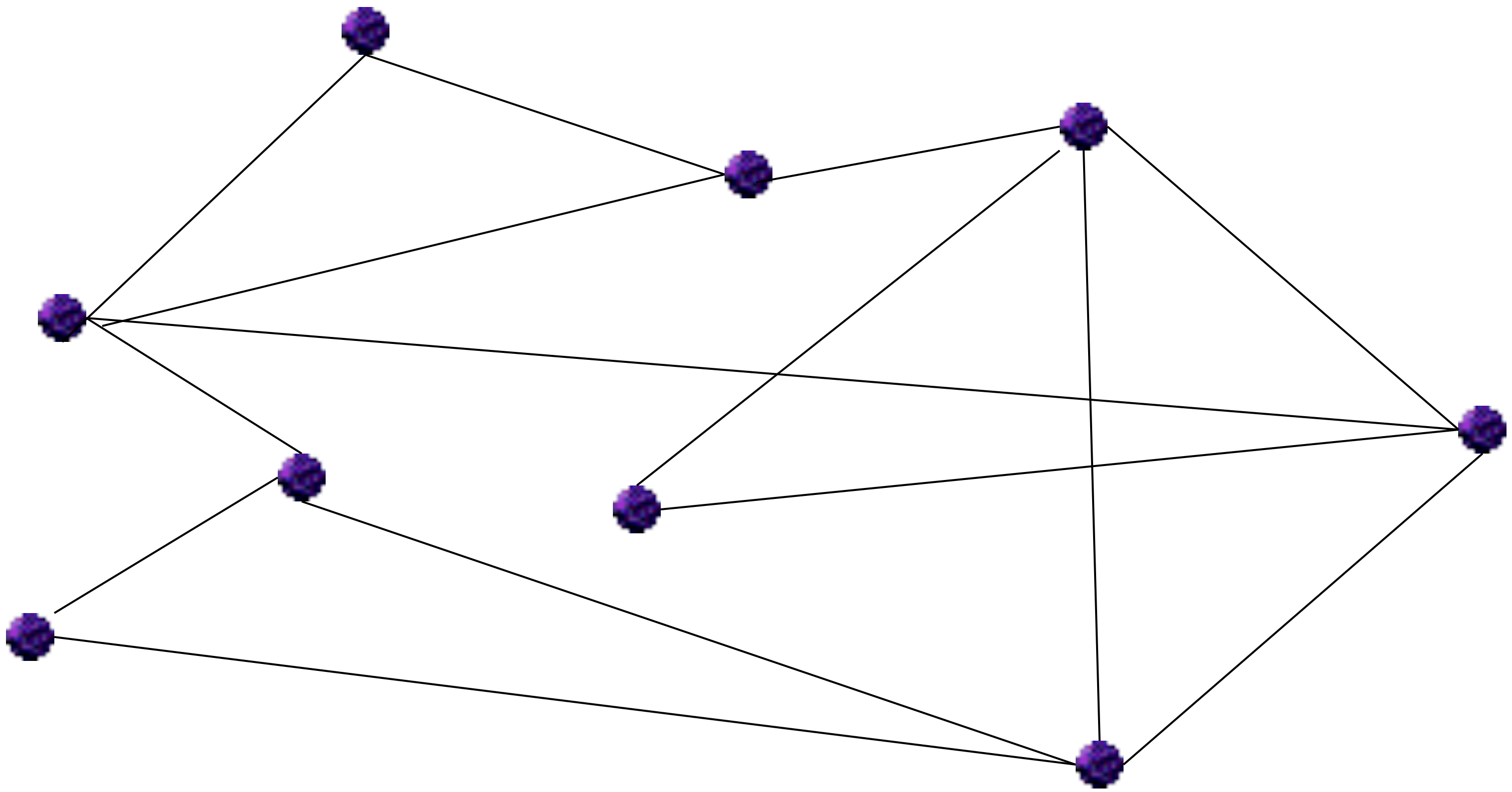
C, Goldreich, Ron, Seshadhri, Sohler, Shapira ’12

Testing cycle-freeness can be done in $O^*(\sqrt{n})$:

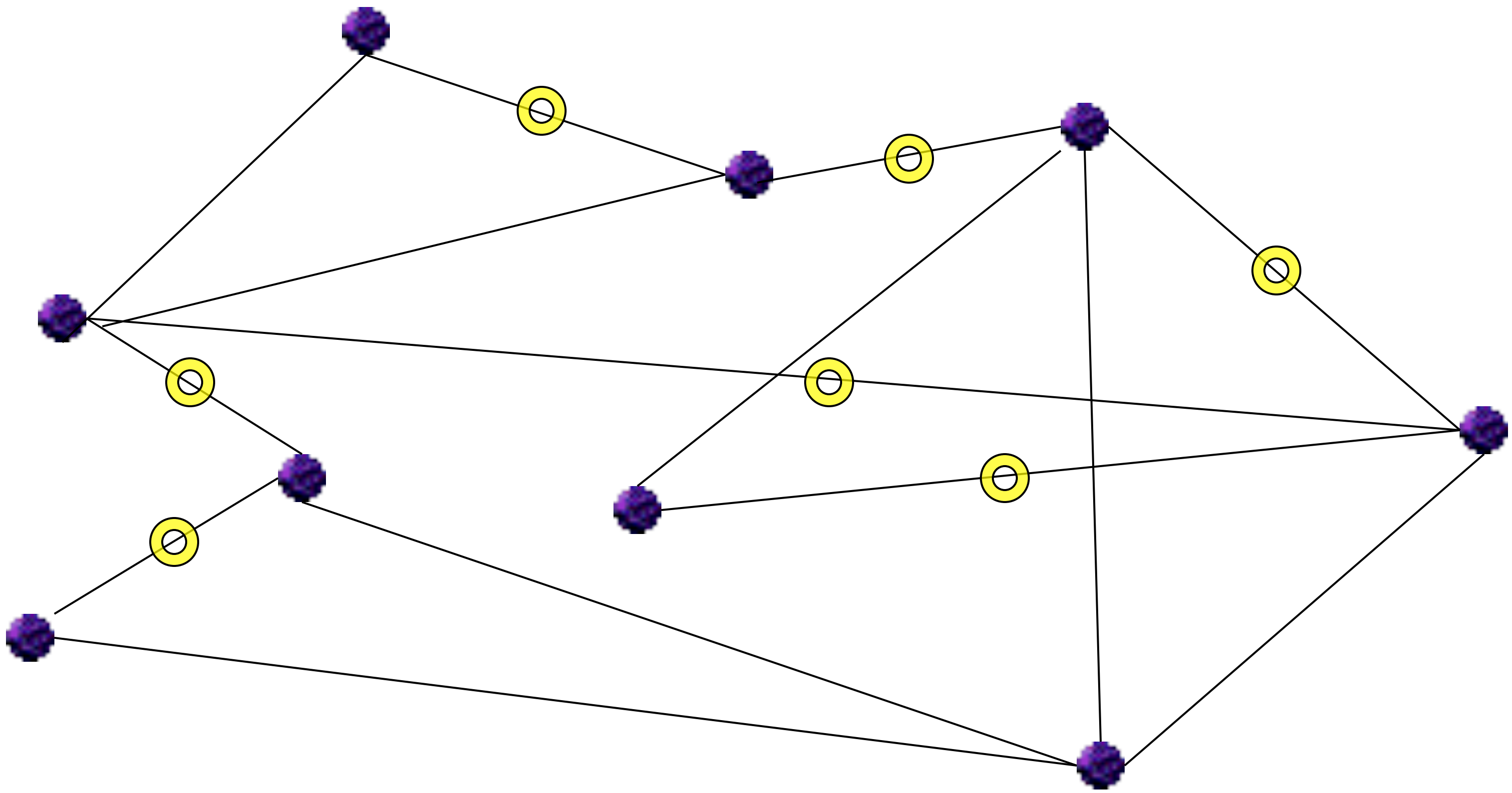
- we know how to test bipartiteness (no odd-length cycles)
- reduce testing cycle-freeness to that of bipartiteness



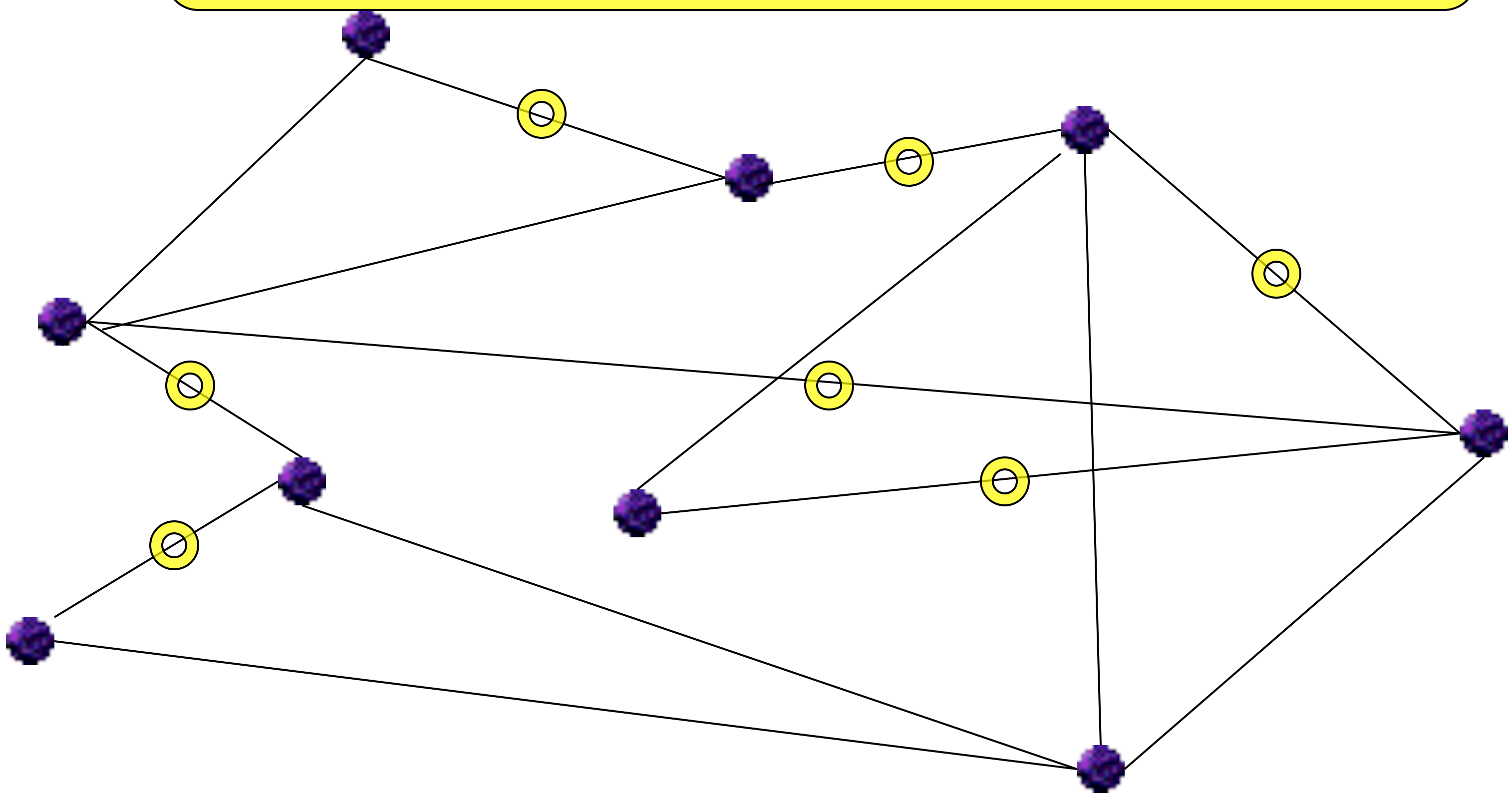
Idea: original graph G has lots of cycles iff new graph $G{\uparrow}*$ has lots of odd-length cycles



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 ε-far from cycle-free ⇔
obtained graph is $\Theta(\varepsilon)$-far from bipartite

Testing cycle-freeness in bounded-degree graphs one-sided error

C, Goldreich, Ron, Seshadhri, Sohler, Shapira ’12

Testing cycle-freeness can be done in $O^*(\sqrt{n})$:

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

Bounded-degree adjacency list model

C, Goldreich, Ron, Seshadhri, Sohler, Shapira '12

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 $O^*(\sqrt{n})$

If G is ε-far from C_k-minor-freeness then we can find a cycle of length $O(\varepsilon^{-1}\log n)$ in $O^*(\sqrt{n})$ time

Bounded-degree adjacency list model

C, Goldreich, Ron, Seshadhri, Sohler, Shapira ’12

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

Can we do better?

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

Goldreich, Ron’02 proved it for $H = C_2$

Bounded-degree adjacency list model

C, Goldreich, Ron, Seshadhri, Sohler, Shapira '12

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

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

For any fixed tree T, testing minor T-freeness with one-sided error can be done in constant-time

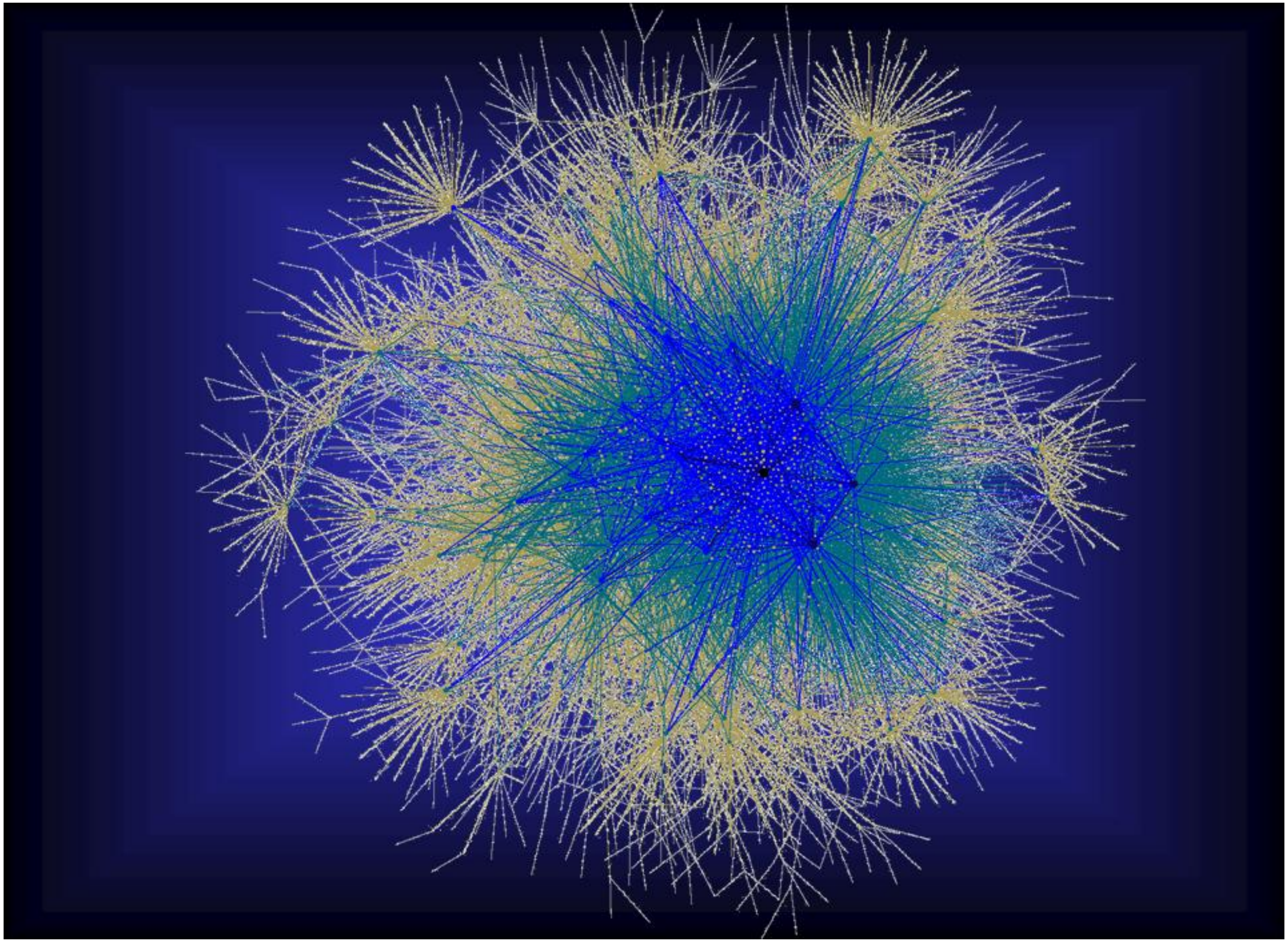
Bounded-degree adjacency list model

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 ε
if and only if
H **is cycle-free**

Testing H-minor-freeness for H having a cycle needs time $\Omega(\sqrt{n})$, but we don't have any further good complexity characterization

Testing planarity



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 ε-far from planar

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

For each subgraph of constant size,
check the number of its occurrences in G
No all frequencies are possible in planar graphs!

Testing planarity

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

- Runtime: $2 \uparrow 2 \uparrow 2 \uparrow poly(1/\varepsilon)$
- Hassidim et al.'09 improved the runtime to $2^{\mathrm{poly}(1/\varepsilon)}$
  - 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: $2\uparrow 2\uparrow 2\uparrow poly(1/\varepsilon)$
- Hassidim et al.'09 improved the runtime to $2^{\mathrm{poly}(1/\varepsilon)}$
  - with somewhat simpler analysis and simpler algorithm
- Levi and Ron'13 improved the runtime to $2\uparrow O(\log\uparrow 2\,(1/\varepsilon))$

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]

- Runtime: $2^{O(\log^2 (1/\varepsilon))}$ (constant for $\varepsilon = O(1)$; still superpolynomial in ε)

- 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

Extension: all minor-closed properties

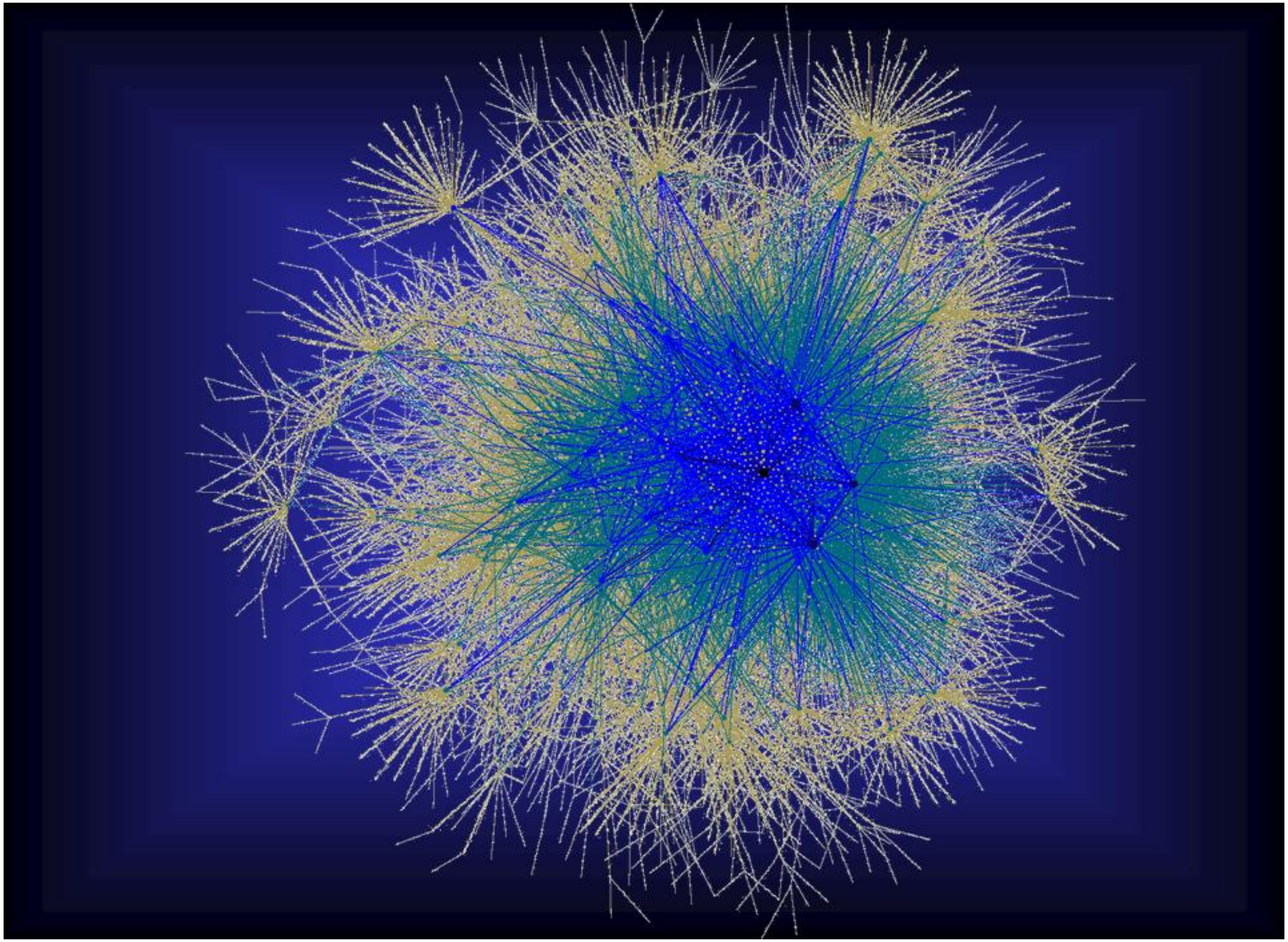
- Every minor-closed property can be tested in a similar way
- Minor-closed properties include:
  - Planar,
  - Outer-planar,
  - Series-parallel,
  - Bounded-genus,
  - bounded tree-width,
  - ...

- 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

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-node
    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 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

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

Testing in planar graphs

- One can make this idea to work to design property testers for planar graphs (of constant max-degree) **for all hereditary properties**
- Key property: every hereditary property can be characterized by a set of minimal forbidden induced subgraphs
- Hence: we only have to check if these subgraphs don't exist in small components

Testing in hyperfinite graphs

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

- Works for all "**non-expanding families**" of graphs (class of **hyperfinite graphs**)

- For every hereditary property P, for any "non-expanding" bounded-degree graph G,

  testing if G has P can be done in constant-time

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)

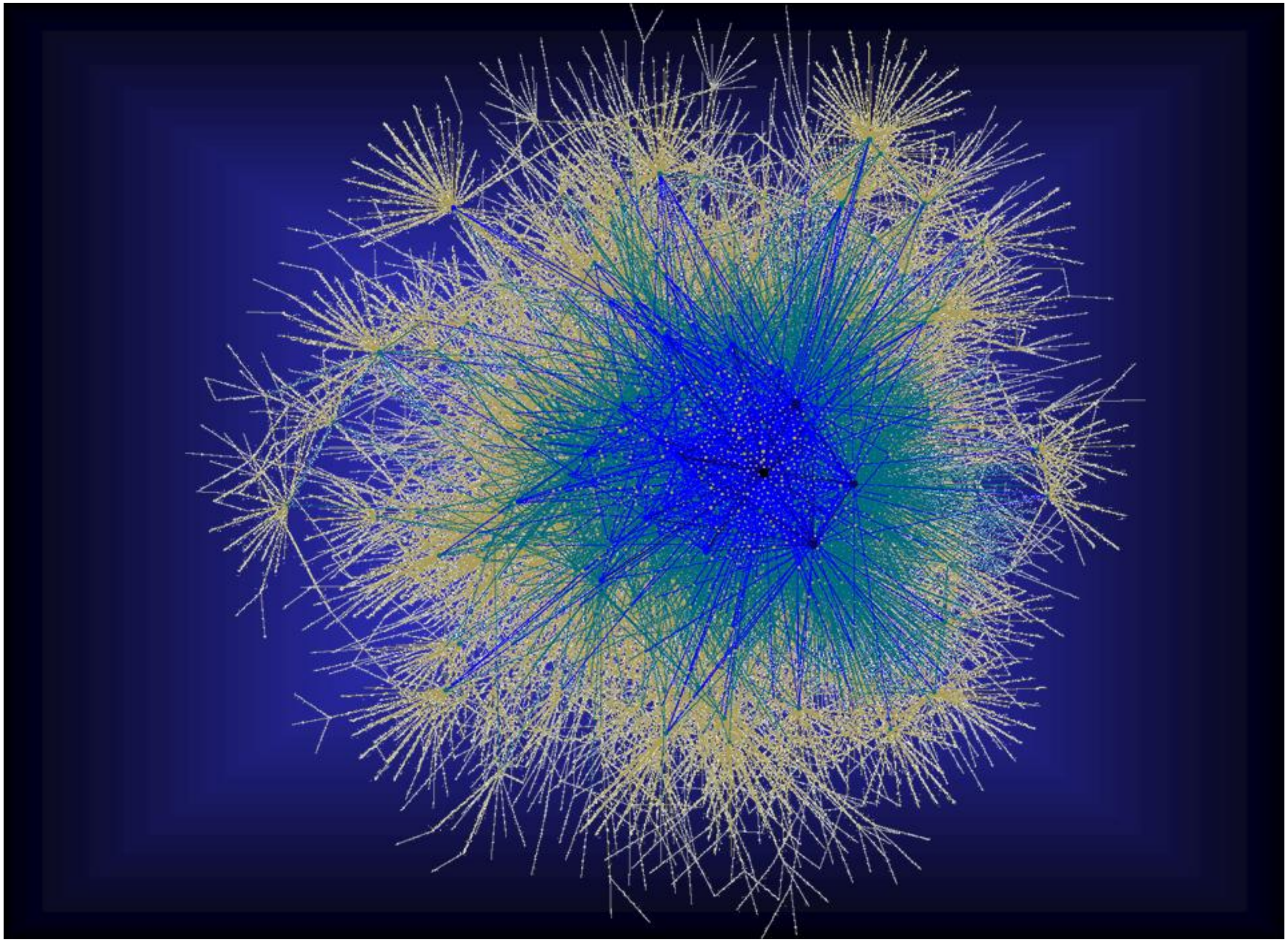
These techniques don’t work for arbitrary-degree graphs

Testing planarity in arbitrary degree graphs requires $\Omega(\sqrt{n})$ time

Two instances:

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

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

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

Access to G via oracle:
Return a random neighbor of v

Access to G via oracle:
Return the ith neighbor of v

Return the degree of v

Testing in arbitrary graphs

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

➔ Graph exploration is expensive

Testing in arbitrary planar graphs

- C, Monemizadeh, Onak, Sohler (2011)

- 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

Extensions

- Broader class of graphs than planar
  - Graphs defined by arbitrary fixed forbidden minors
- Extension beyond bipartiteness: work in progress

Summary

- Big Graphs need good understanding of testing algorithms

- Many graph properties can be tested efficiently
  - Sometimes in constant-time
  - More often in sublinear-time

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

Summary

- 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
    - Very little is known

- What haven’t been mentioned:
  - Directed graphs
  - Access through random edges