Property Testing for Sparse Graphs: Structural graph theory meets Property testing

Ken-ichi Kawarabayashi
National Institute of Informatics(NII)&
JST, ERATO, Kawarabayashi Large Graph Project

Joint work with Yuichi Yoshida(NII).

STOC’13

Purpose of this talk

- How Structural graph theory helps property testing?

- Warning: I am NOT an expert on property testing..

Contents

- What is the property testing?
- Dense graphs model.
- Bounded degree graphs with separators.
- Bounded degree graphs with no separators
  our main contribution
- Tools from property testing and graph minors
- Summary

Property Testing

- Dense Graph Model:

Connected to Szemeredi's Regularity Lemma

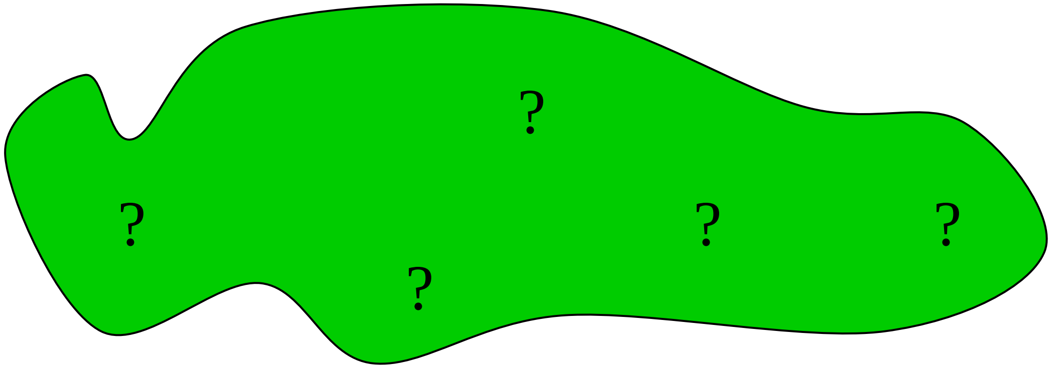
(Due to Alon et al.)

- Bounded Degree Model:

Connected to Structural Graph Theory and Graph Minor (from this work!)

Property Testing (Informal Definition)

For a fixed property P and any object O,
determine whether O has property P,
or whether O is far from having property P
(i.e., far from any other object having P).



Task should be performed by querying the object (in as few places as possible. Sublinear or even constant time).

Examples

- The object can be a graph (represented by its adjacency matrix), and the property can be 3-colorabilty.
- The object can be a string and the property can be membership in a given regular language L.
- The object can be a function and the property can be linearity.

When can Property Testing be Useful?

- Object is to too large to even fully scan, so must make approximate decision.
- Object is not too large but
  (1) Exact decision is NP-hard (e.g. coloring)
  (2) Prefer sub-linear approximate algorithm to polynomial exact algorithm.

Actual Computation Results for the Shortest Paths Problem Using High-Performance Computer (HPC)(2011)

Based on Dijkstra's algorithm (Running time: O(n log n))

- *Graph of the entire United States (n=24,000,000 points, 58,000,000 edges)*: **3 seconds**
- *Very large scale graph (n=*10^9 *points, 2* $\times 10^9$ *edges)*: **870s**

Individual personal computers need >1000 times！

We cannot use Dijkstra's algorithm！

Graph Property Testing

Very general setting:

P = graph property to test

(*k*-colorability, planarity, non-existence of a copy of *H*, etc.)

Input: graph *G* on *n* vertices, n→∞

Promise: $G \in P$ (positive)

or: *G* is ε-far from *P* (negative)

(More than ε-percentage of description of *G* should be changed to get $G \in P$)

Edge Addition or Edge Deletion

Algorithm(typically randomized): Constant time(Sublinear)

$G \in P \Rightarrow$ Pr[*A* accepts *G*] ≥ 2/3

G is ε-far from $P \Rightarrow$ Pr[*A* rejects *G*] ≥ 2/3

two-sided error

$G \in P$, Pr[*A* accepts *G*] =1 – one-sided error algorithm

Contents

- What is the property testing?
- Dense graphs model
- Bounded degree graphs with separators
- Bounded degree graphs with no separators

  our main contribution
- Tools from property testing and graph minors
- Summary

Property Testing in Dense Graphs

- Formally defined in GGR'98

(appeared implicitly in combinatorial papers in 70's, 80's)

Input graph description: adjacency matrix G=(V,E), V=[n]

$$A_{n\times n} \qquad a_{ij} = \begin{cases} 1, & (i,j)\in E(G) \\ 0, & otherwise \end{cases}$$

Algorithm: queries the adjacency matrix of G

Want: Constant-time query!

Distance: *G* is is *ε*-far from *P* if $\geq \varepsilon n^2$ entries in *A(G)* need to be changed to get $G \in P$ *(addition or deletion)*

Property Testing in Dense Graphs – Brief Summary

"… It's all about REGULARITY." (Alon, Fischer, Newman and Shapira'06)

Every ``heredity property(closed under deletion)" is constant-time testable if and only if there is a "Szemeredi partition".

- Very strong (and fruitful) connection between property testing in dense graphs and the Szemerédi Regularity Lemma and its versions

Dense Graph Model - limitations

- Suitable/tailored for dense graphs only

- Degenerate for many graph properties

Ex. : P = " G is connected"

- Always answer "YES"

(Imagine edge addition: $dist(G,P) \leq n-1 \ll \varepsilon n^2$)

Property Testing

- Dense Graph Model:

Connected to Szemeredi's Regularity Lemma

(Due to Alon et al.)

- Bounded Degree Model:

Connected to Structural Graph Theory and Graph Minor (from this work!)

Contents

- What is the property testing?
- Dense graphs model
- Bounded degree graphs with separator
- Bounded degree graphs with no separators: our main contribution
- Tools from property testing and graph minors
- Summary

Property Testing in Bounded Degree Graphs

Introduced by Goldreich and Ron'97(GR97)

- **Assumption**: max degree of an input graph $G \leq d$=*constant*, $\varepsilon \ll 1/d$
- **Graph representation**: by incidence lists

  $L(v_i)=(v_{i,1},\ldots,v_{i,d})$ - list of neighbors of v_i
- **Distance:** G is ε-far from P if need $\geq \varepsilon dn$ modifications in incidence lists to get $H \in P$

  (addition or deletion)

Bounded Degree Graphs - an Example

Th. (GR'97): Connectivity in bounded degree model can be tested in $O(1/\varepsilon^2)$ queries

Proof: Assume: G is ε-far from being connected

⇓

G has $\geq \varepsilon n$ connected components

⇓

G has $\geq \varepsilon n/2$ *con.* components of size $\leq 2/\varepsilon$ (= small components)

⇓

$\geq \varepsilon/2$ percentage of all vertices in small components

Property Testing in Bounded Degree Graphs

Algorithm: Repeat $O(1/\varepsilon)$ times:

1. Sample a random vertex $v \in_R V$
2. Explore the connected component $C(v)$ of v till accumulate $2/\varepsilon$ vertices
3. If $|C(v)| \leq 2/\varepsilon$ – reject

   (G is ε-far from being connected)

If never reject – accept

One-sided error algorithm with complexity

$O(1/\varepsilon^2)$

More careful analysis $\tilde{O}(1/\varepsilon)$ queries

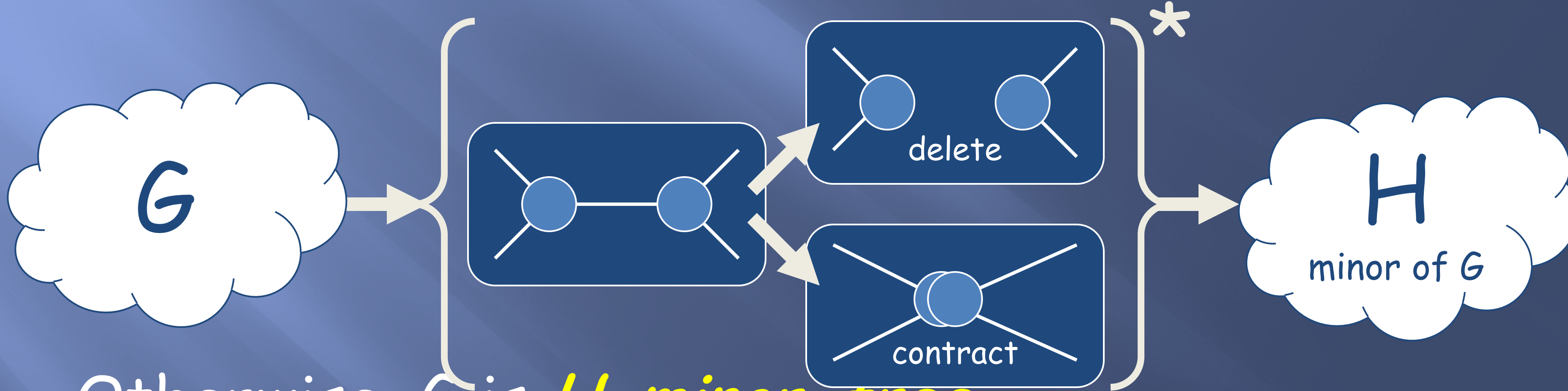
Three reasons of Constant-time testability in bounded-degree model

Properties	Why is it testable?
$\triangle$–freeness, H–freeness [GR02]	Locally determined
k–edge–connectivity [GR02] k edge–disjoint spanning trees [ITY'12]	Edge–augmentation / matroid theory.
Planarity, H–minor–freeness [BSS08, HKNO09]	Existence of separators

Is there any other kind of testable properties?

Graph Minors

- A graph G *has a minor H* if
  H can be formed by removing and contracting edges of G



- Otherwise, G is *H-minor-free.*
- *Minor-closed: Closed under minor operations.*
- For example, Planar graphs are minor-closed.
- Kuratowski's theorem

H-minor-free in Constant-time testing(BSS08)

Can figure out

G has no εdn edges (or εn vertices) X such that G-X has no H-minor (or is nonplanar).

in constant time!

Three reasons of testability in bounded-degree model

Properties	Why is it testable?
△-freeness, H-freeness [GR02]	Locally determined
k-edge-connectivity [GR02] k edge-disjoint spanning trees [ITY'12]	Edge-augmentation / matroid theory.
Planarity, H-minor-freeness [BSS08, HKNO09]	Existence of separators

Is there any other kind of testable properties?

Separator

- Given a graph G, if V(G) can be partitioned into three parts A,B,C such that

1. there is no edge between A and B, and
2. $|G|/3 <= |A|,|B| < 2|G|/3$,

Then C is called a separator .

We are interested in a separator of SMALL order, i.e, sublinear order.

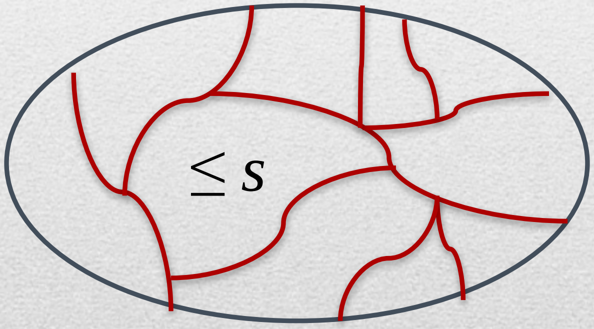
Separator Theorem: Every H-minor-free graph has a separator of order o(n).

Using separators: Decomposition lemma

Consider a H-minor-free graph G.

$\forall\, |H|$ and ε, $\exists\, s$ s.t. we can decompose G by removing εn edges into component of size $\leq s$,

H–minor–free:



# of edges crossing —— is $\leq \varepsilon dn$

Sketch for H-minor-free Constant-time testing(BSS08)

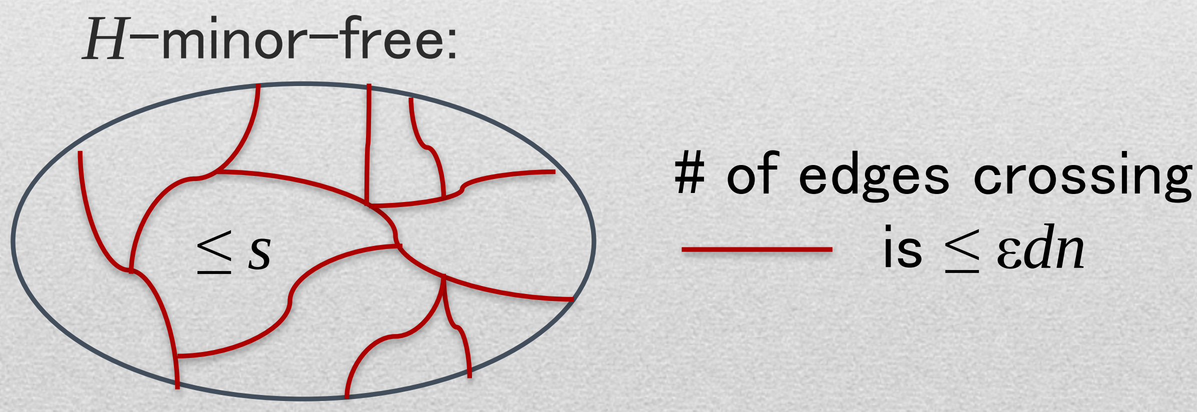
- Structural graph theory approach

  Using separators, decompose H-minor-free graphs into small graphs (easily follows from separators.)

- Partitioning oracle (Tools from Property testing)

Using Decomposition thm: Partitioning oracle

- It suffices if we can access the graph G' given by the decomposition lemma. How??
- Partitioning oracle provide query access to a decomposition, designed for H–minor–free graphs. [HKNO09]



Keys for H-minor-free testing(BSS08)

Need to combine Structure graph theory and Property testing!

- Structural graph theory approach

Using separators, decompose H-minor-free graphs into small graphs.

⇒ Easy.

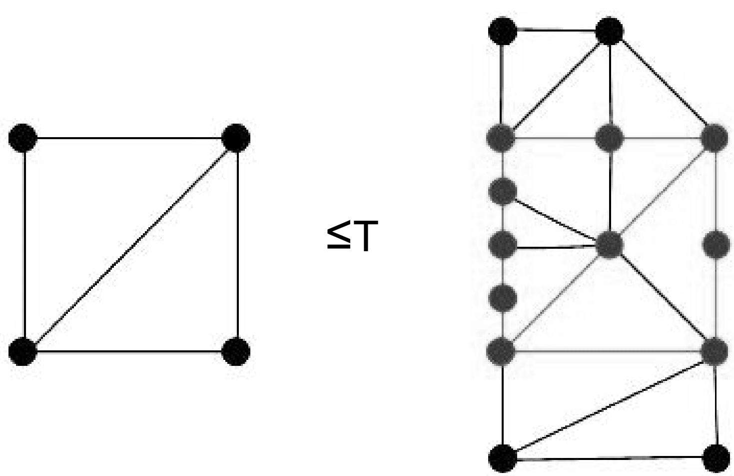
- Partitioning oracle (Tools from Property testing)

⇒ Main Task

How about subdivision-free?

Subdivision of a graph: replacing each edge by a path of length 1 or more.

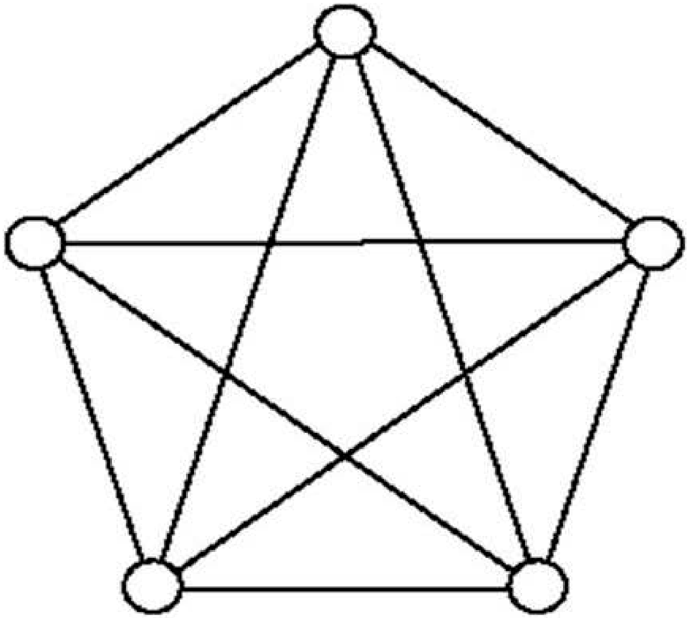
G contains a subdivision of H if G contains a subgraph H' that is a subdivision of H.

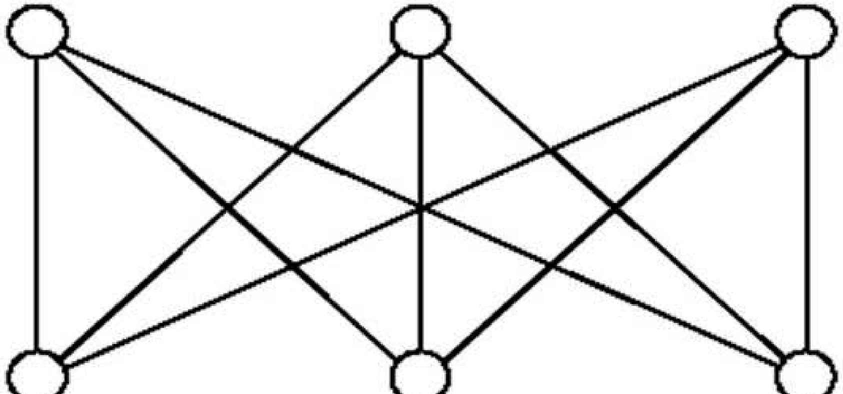


Branch Vertices: vertices of H that correspond to "vertices" (not in a path of length 1 or more)

Kuratowski's Theorem Ver 2

- A graph is planar (can be embedded in a plane without edge crossings) if and only if it contains neither K_5 nor $K_{3,3}$ as a





Main contribution

K_t–subdivision–freeness is constant–time testable for any $t \geq 1$.

Can figure out

G has no *εdn* edges (or *εn* vertices) *X such that G-X* has no K_t–subdivision.

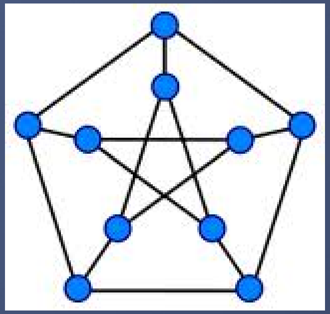
in constant time!

Main contribution

K_t–subdivision–freeness is constant–time testable for any $t \geq 1$.

- Not locally determined
- Nothing to do with edge-augmentation / matroids.
- May not have separators
  - an expander graph with max degree t-2.
- First Property that can contain an expander!

Expander Graph



- ***Intuitively:*** *a graph for which any "small" subset of vertices has a relatively "large" neighborhood.*
- *Hence no separator of order o(n).*
- *Can be defined in Algebraic sense and in Probabilistic sense too!*
- ***Property***: It behaves like a (sparse) random graph!
- Used many areas in Math and CS!

Contents

- What is the property testing?
- Dense graphs model
- Bounded degree graphs with separators
- Bounded degree graphs with no separators

  our main contribution
- Tools from property testing and graph minors
- Summary

Proof Sketch

Reminder: Sketch for H-minor-free testing

Need to combine the two approaches

- Structural graph theory approach

Using separators, decompose H-minor-free graphs into small graphs (easily follows from separators).

- Partitioning oracle (Tools from Property testing)

⇒ Main Task

Warning: No separator for the subdivision case.

So decomposition thm for subdivision case is not trivial. Need "deeper" structural graph approach! (then can combine with property testing)

Testing K_t-subdivision-freeness: High level

Basically following the minor case!

Combinations of Structural graph and Property testing!

Decomposition thm

- Decompose G into components by removing $\varepsilon' n$ edges
  - of constant size, or
  - with large clique minor and no small cut
- Design a tester that works locally given the decomposition.

Constant-time tester for K_t-sub.-freeness

- Use and modify "partitioning oracle" to obtain query access to the decomposition ⇒ Not hard.

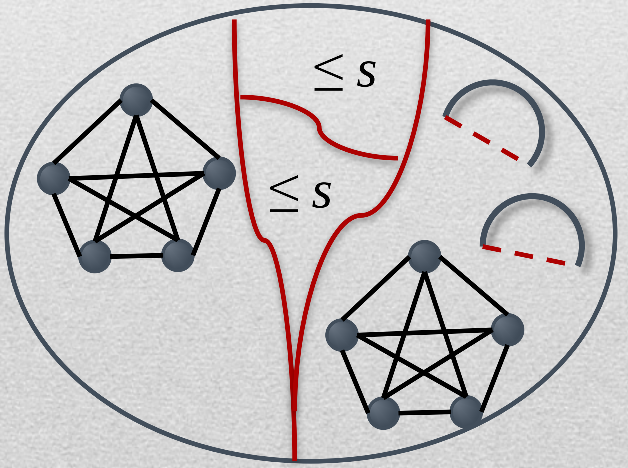
Decomposition lemma

l–hidden cut C: every component in $G - C$ has size at least $l|C|$.

- No separator, but using graph minor(tangle), we have the following!

$\forall t, t'$ and ε, $\exists s$ s.t. we can decompose G by removing εn edges into components

1) of size $\leq s$, or
2) with $K_{t'}$–minor and no $(1/\varepsilon)$–hidden cut of size $< t - 1$.



Using Decomposition lemma

Decompose G by removing $\varepsilon' dn \ll \varepsilon dn$ edges into

1) small components
2) components with $K_{t'}$–minor and no hidden cut of size $< t - 1$.

It suffices to test the resulting graph G'(after removing edges).

- If G is K_t–sub.–free ⇒ G' is K_t–sub.–free
- If G is ε–far ⇒ G' is $(\varepsilon\text{-}\varepsilon')$–far

Our algorithm, at a high level

Suppose that we can access the decomposition!

1) **small components**

- easy to test (exactly same as the minor case)

Need to look at the following case!

2) large components with K_t-minor and no l-hidden cut of size $< t-1$.

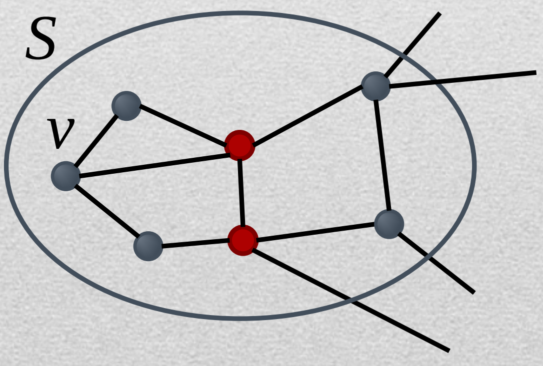
- Estimate # of dangerous vertices w.r.t. small neighborhood and accept if it is $< \varepsilon n/4$.
- Can be done in constant time.

Dangerous

A vertex v is dangerous w.r.t. $S \subseteq V$ if v is not separated in S by a cut of size $< t-1$.

- We cannot exclude the possibility that v is a branch of K_t–subdivision.

Ex.



v is not dangerous w.r.t. K_4 because of the red cut.

Correctness

If G' is ε–far:

- Many ($\geq \varepsilon n/2$) dangerous vertices as otherwise we can remove edges incident to them.

If G' is K_t–subdivision–free.

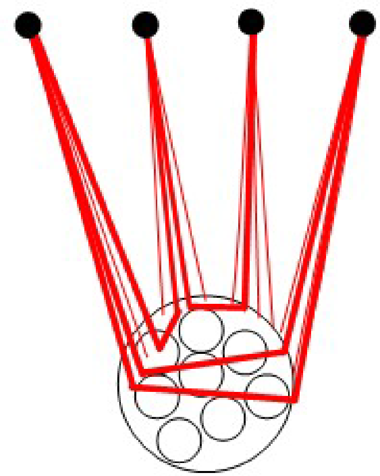
- Want to show there are a few ($\leq \varepsilon n/1000$) dangerous vertices.
- How many dangerous vertices can a large component have? Use tools from Graph Minor!

Contents

- What is the property testing?
- Dense graphs model.
- Bounded degree graphs with separator
- Bounded degree graphs with no separators:
  our main contribution
- Tools from property testing and graph minors.
- Summary

Tools from graph minors

Suppose that there is a set S of |V (H)| vertices that are very far (only depending on |V(H)|) from each other, and each having degree > |V(H)|.
Suppose there is a large clique minor.



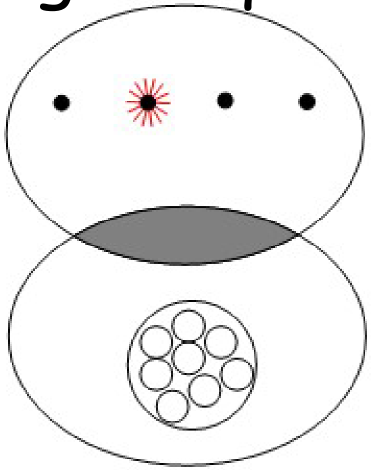
Graph Minor tells only
Two possibilities:

(1) There are many disjoint paths from S to the clique minor

⇒ Using the clique minor as a crossbar, we can complete the paths into a H-subdivision
Winning!

Tools from graph minors

Suppose that there is a set S of |V (H)| vertices that are *very far* (only depending on |V(H)|) from each other, and each having degree > |V(H)|.
Suppose there is a large clique minor.



Graph Minor tells only Two possibilities:

(2) There is a small separator between S and the clique minor

Remember!
Big Piece: 1 . More than constant size.
2. No "hidden" cut.
3. contains a large clique minor.

So (2) does not happen! So small # of dangerous vertices!

Correctness

If G' is K_t-sub-free.

- Each large component has c dangerous vertices.
- There are at most n / s large components.
- Thus, there are at most $cn / s \ll \varepsilon n/1000$ dangerous vertices.
- Remaining task:

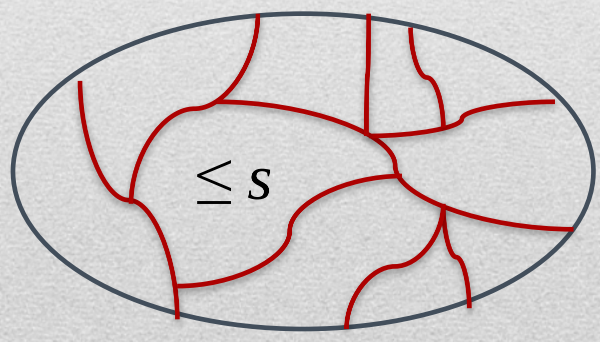
How to access the decomposition??

Last step: How to access the decomposition?
Constant-time tester

Reminder: Partitioning oracle

- Partitioning oracle provide query access to a decomposition, originally designed for H–minor–free grahps. [HKNO09]

H–minor–free:



# of edges crossing —— is $\leq \varepsilon dn$

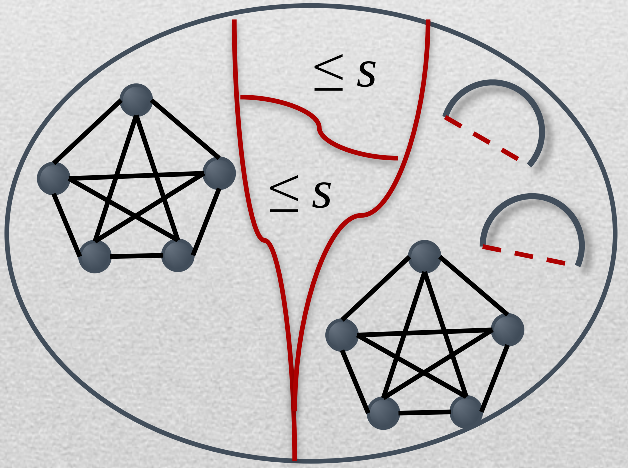
Reminder: Decomposition lemma for subdivision

l–hidden cut C: every component in $G - C$ has size at least $l|C|$.

- Hard for local algorithms to detect

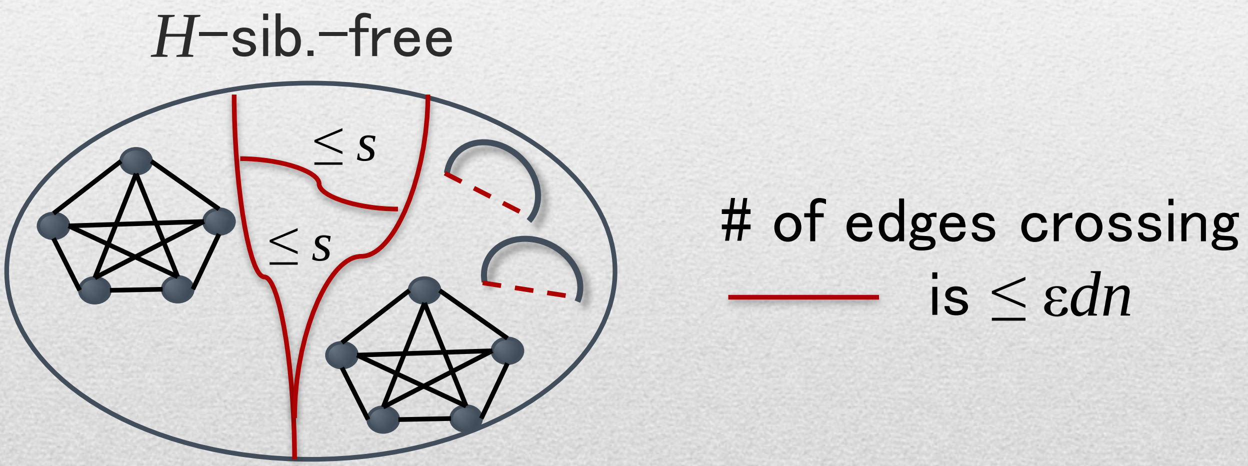
$\forall t, t'$ and ε, $\exists s$ s.t. we can decompose G by removing εn edges into components

1) of size $\leq s$, or
2) with $K_{t'}$–minor and no $(1/\varepsilon)$–hidden cut of size $< t - 1$.



Modified Partitioning oracle

Modify [HKNO09] to give query access to G' for K_t–sub.–free graph. (not hard)



Though we have a little error, it does not affect the # of dangerous vertices too much.

Conclusions

Main result:

K_t–sub.–freeness is constant–time testable.

Structure Graph Theory: Decomposition

Property Testing: Accessing the decomposition

Nice combination of Structural graph theory and Property testing!

Previously, property testing is harder, but in our case, structural part is harder!

Property Testing

- Dense Graph Model:

  Connected to Szemeredi's Regularity Lemma

(Due to Alon et al.)

- Bounded Degree Model:

Connected to Structural Graph Theory and Graph Minor (from this work!)

Future work

Open problems:

- Query complexity: $2^{\wedge}(d^{\wedge}\mathrm{poly}(\varepsilon/2^{\mathrm{poly}(t)}))$.
- Can we test H–(topological–)minor–freeness in adjacency list model?
- Some other classes? (Immersion is done by this work, but what else?)

Thank you for your attention!

Any Question?

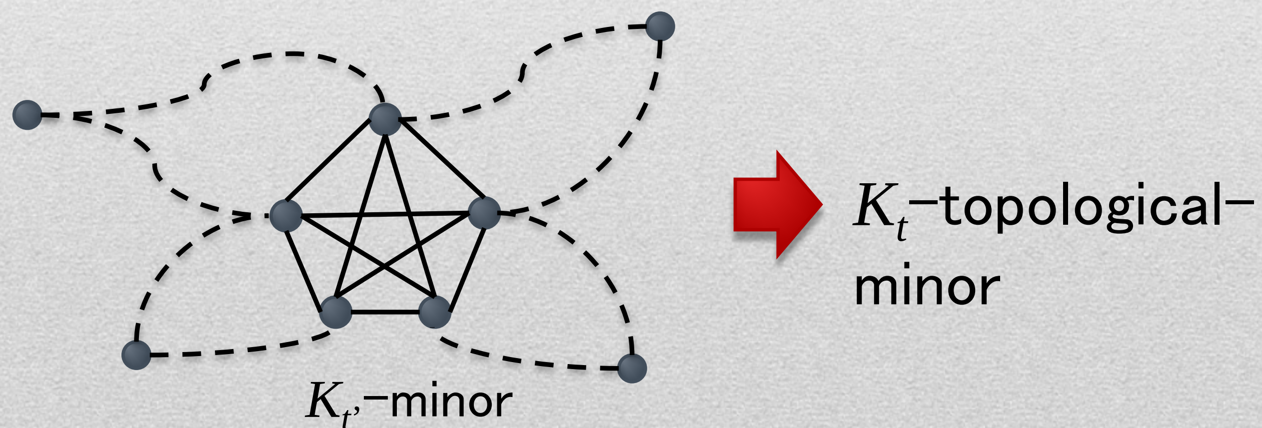
Many Thanks !

A sufficient condition to have K_t–tm

$\forall t$ and l, $\exists t'$, c, and r such that

- $K_{t'}$–minor
- no l–hidden cut of size $< t - 1$.
- $\geq c$ dangerous vertices w.r.t. radius–r balls

$\Rightarrow$ K_t–topological–minor



Main Tools

Th. *P* = "*G* is K_t-subdivision-free"

P can be tested in time $O_\varepsilon(1)$ in bounded degree graphs by a 2-sided error algorithm.

Main Tools

1. Extension of partitioning oracle(correctness based on graph minor)
2. Tools from graph minor!