Sublinear Time Approximation Algorithms (Continued):

Estimating Size of Maximal Matching in degree bounded graph:

Why?

1) Vertex Cover (VC) = Maximal Matching size
   a) VC ≤ 2 MM
      ← put all nodes in MM into VC. Covers all edges if not violated maximality.
   b) in d≤d graphs, maximal matching is of size \( \frac{N}{d} \)
      ← see this by implementing greedy algorithm

Greedy Sequential Matching Algorithm:

\[ M = \emptyset \]
\[ \forall e \neq (u,v) \in E \]
\[ \text{if neither } u \text{ or } v \text{ matched}
\]
\[ \text{add } e \text{ to } M \]

Output \( M \)

Observe:

\( M \) maximal since if \( e \notin M \), either \( u \) or \( v \) already matched earlier.
Oracle Reduction Algorithm

- \( S' \leq s = \frac{2}{\varepsilon^2} \) nodes chosen iid

- For all \( v \in S' \),
  \[ X_v = \begin{cases} 
  1 & \text{if any call to oracle } O((v, w)) \text{ for } w \in N(v) \text{ returns "yes"} \\
  0 & \text{otherwise} 
  \end{cases} \]

- Output \( n \leq 2s \sum_{v \in S'} X_v + \frac{\varepsilon}{2} n \)

Why does it work?

\[
|M| = \frac{1}{2} \sum_{v \in V} X_v
\]

\[
E[|\text{output}|] = E[\frac{n}{2s} \sum_{v \in S'} X_v] + \frac{\varepsilon}{2} n \]

\[
= \frac{n}{2s} \sum_{v \in S'} E[X_v] + \frac{\varepsilon}{2} n \]

\[
= \frac{n}{2s} \times |V| \cdot 2 \times \frac{|M|}{|V|} \times \frac{1}{2} n = |M| + \frac{\varepsilon}{2} n
\]

\[
Pr \left[ \frac{n}{2s} \sum_{v \in S'} X_v + \frac{\varepsilon}{2} n - E[|\text{output}|] \geq \frac{\varepsilon}{2} n \right] \leq \frac{1}{3} \text{ by additive Chernoff-Hoeffding }
\]

\[
Pr \left[ \frac{n}{s} \sum_{v \in S'} X_v - |M| \geq \frac{\varepsilon}{2} n \right] \leq \frac{1}{3}
\]
How to implement the oracle?

Main idea: figure out
"what would greedy do on \((v_i, w)\)?"

how can you do this in sublinear time?

Big Problem: Greedy is very "sequential"
i.e. can have long dependency chains

ex. \[ 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \cdots \rightarrow 212 \]

is this edge odd or even in edge order?

How to implement oracle based on greedy?

To decide if \(e\) in matching, adjacent
need to know decisions for edges that came before \(e\) in ordering

Do not need to know anything about any edge that comes after \(e\) or in the ordering by greedy before \(e\)

ie. if any edge that comes before \(e\) in ordering is matched
then \(e\) not matched, else \(e\) is matched
How to break length of dependency chains:

assign random ordering to edges

e.g.,

Example

Start with 0.5
- recurse on 0.3
  - recurse on 1 - in M
    (don't look at 0.7)
    so 0.3 not in M
    - don't look at 0.7
  - recurse on 0.4
    - recurse on 0.2
      - don't look at 0.8
    - 0.2 in M
    - 0.4 not in M
    - 0.5 in M
Implementation of oracle:

To check if \( e \in M \):

\[\forall e' \text{ neighboring } e,\]

* if \( r_e < r_{e'} \), recursively check \( e' \)
* if \( e' \in M \), return "\( e \in M \)" and halt

else continue

return "\( e \in M \)" (since no \( e' \) nbr of \( e \) with \( r_e < r_{e'} \) found with \( e' \in M \))

Correctness:

follows from correctness of greedy

Complexity:

Claim: expected \# queries to graph per oracle query is \( 2^{O(d)} \)

Claim \( \Rightarrow \) total query complexity is \( \frac{2^{O(d)}}{\varepsilon^2} \)
Pf of Claim

* Consider a tree where the root node is labelled by the original query edge, and children of each node are edges adjacent to it.

* Will only query paths that are monotone decreasing in rank.

\[ \Pr [ \text{given path of length k explored}] = \frac{1}{(k+1)!} \]

\( \leq d^k \) nodes at dist k in tree

\[ E [\text{# nodes explored at distance k}] \leq \frac{d^k}{(k+1)!} \]

\[ E [\text{total # nodes explored}] \leq \sum_{k=0}^{\infty} \frac{d^k}{(k+1)!} \leq e^d \]

\[ E [\text{query complexity}] \leq d \cdot e^d = e^d = 2^{o(d)} \]
Testing $H$-minor freeness

all graphs have degree $\leq d$

defs. $H$ is "minor" of $G$
if can obtain $H$ from $G$ via
vertex removals, edge removals, edge contractions

$G$ is "$H$-minor-free" if $H$ not a minor of $G$

$G$ is "$\epsilon$-close to $H$-minor-free" if

can remove $\leq \epsilon|V|n$ edges to make it
$H$-minor-free

minor closed property $P$:

if $G \in P$ then all minors of $G$ are in $P$

Cool theorem [Robertson-Seymour]
every minor-closed property is expressible
as a constant # of excluded minors

Today's goal: testing $H$-minor free graphs

is $T$ passes $H$-minor free graphs
fails $\epsilon$-far from $H$-minor free $G$ with failure probability $\leq \epsilon/4$
more details:

- $G$ is "$(\varepsilon, k)$-hyperfinite" if
  - can remove $\leq \varepsilon n$ edges
  - and remain with connected components of size $\leq k$

- $G$ is "$\rho$-hyperfinite" if
  - $\forall \varepsilon > 0$, $G$ is $(\varepsilon, \rho(\varepsilon))$-hyperfinite

Useful theorem

Given $H$

- $\exists C_H$ s.t. $\forall \varepsilon \in (0, 1)$, every $H$-minor free graph of $\text{deg} \leq d$
  - is $(\varepsilon d, C_H \varepsilon^2)$-hyperfinite

(i.e., remove $\leq \varepsilon d n$ edges
  - and components are $O(\varepsilon^2)$)

Note: Subgraphs of $H$-minor free graphs also $H$-minor free

- so also hyperfinite
  - but only remove edges in proportion to # nodes in subgraph!


Why is hyperfiniteness useful?

- Partition graph $G$ into $G'$ only small (const size) components remain
  - removing few edges
  - if can't do this, $G$ is not $H$-minor free
- if $G'$ is close to having property, so is $G$
- to test $G'$ by picking random components
  + seeing if they have property

Assume have "partition oracle" $P$: (with parameters $\frac{\varepsilon d_{\bar{g}}}{q}, k$)

**Input:** vertex $v$

**Output:** $P([v])$ $v$'s partition name

- $\forall v \in V: (1) |P([v])| \leq k$
  - (2) $P([v])$ connected
- if $G$ is $H$-minor free
  - with prob $= \frac{9}{10}$
  - $|E(u, v) \in E : P(u) \neq P(v)| \leq \varepsilon d_{\bar{g}}$
Algorithm given partition oracle $P$:

- estimate number $\hat{t}$ of edges $(u,v)$
  
  \[
  \text{st. } P[u] \neq P[v] \text{ to additive error } \leq \frac{\varepsilon dn}{8} \text{ with prob failure } \leq \frac{1}{10}
  \]

- if $\hat{t} \geq \frac{3}{8} \varepsilon dn$ output "far" + halt

- else, choose $S = O(1/\varepsilon)$ random nodes
  
  if for any $s \in S$
    
    $P[s]$ not $H$-minor free, reject + halt

- Accept

Analysis (assuming $P$ always correct)

if $G$ $H$-minor free:

\[
E[\hat{t}] \leq \frac{\varepsilon dn}{4}
\]

sampling bounds $\Rightarrow \hat{t} \leq \frac{\varepsilon dn}{4} + \frac{\varepsilon dn}{8} = \frac{3}{8} \varepsilon dn$ with prob $\geq \frac{9}{10}$

\[
\forall s \in V, P[s] \text{ is } H \text{-minor free}
\]

if $G$ $\varepsilon$-far from $H$-minor free:

\underline{Case 1} P's output doesn't satisfy \[|\delta(u,v) \cup E : P(u) \neq P(v)| \leq \frac{\varepsilon dn}{4}\]

sampling bounds $\Rightarrow \hat{t} \geq \frac{\varepsilon dn}{4} + \frac{\varepsilon dn}{8} = \frac{3}{8} \varepsilon dn$ with prob $\geq \frac{9}{10}$

$\Rightarrow$ output "far" with prob $\geq \frac{9}{10}$
Case 2: P's output satisfies \( |\{ (u, v) \in E : P(u) \neq P(v) \} | \leq \varepsilon d_n \frac{1}{4} \)

\( G' \leftarrow G \) with "crossing" edges removed

\( \text{(w) s.t. } P(u) \neq P(v) \)

\( G' \) is \( \varepsilon \frac{\varepsilon}{4} \) - far from \( G \), so \( \geq \frac{3\varepsilon}{4} \) - far

from \( H \)-minor free \( \Rightarrow \geq \frac{3\varepsilon}{4} \) fraction of nodes belong to component which is not \( H \)-minor free

\( \square \)
Global Partitioning Algorithm

$\pi_1 \ldots \pi_n \leftarrow \text{random permutation of nodes}$

$P \leftarrow \emptyset$

For $i = 1 \ldots n$ do

if $\pi_i$ is still in graph then

if $E \in (k, \delta)$ - isolated neighborhood of $\pi_i$ in remaining graph then $S \in \text{this nbhd}$

else $S \in \text{else}$

$P \leftarrow P \cup S$

remove $S$ from graph

Most nodes have $(k, \delta)$-isolated nbhds

Lemma: if $G'$ subgraph of $G$ with $\geq 8n$ nodes,

$\leq \frac{\epsilon}{30} |V_{G'}|$ nodes don't have $(p(\frac{\epsilon^2}{1800}), \frac{\epsilon}{30})$-isolated nbhd.

Proof:

- $G'$ is $\delta$-hyperfinite, so $G'$ can also be broken up into components of size $\leq p(\frac{\epsilon^2}{1800})$ by removing few edges

- by Markov's inequality (counting), most components have few "removed" edges adjacent to them
Using lemma \( \Rightarrow \leq \frac{\epsilon d n}{q} \) edges cross with prob \( \geq \frac{q}{10} \)

**Local simulation of oracle:**

- Assign random number \( \epsilon \in (0, 1) \) to \( v \)
  - When first see it, use rank orders to define \( v \)

- To compute \( P[v] \)
  - Recursively compute \( P[w] \) for \( w \) within distance \( \leq 2k \) of \( v \)
  - If \( \exists w \in P[w] \) then \( P[v] = P[w] \)
  - Else look for \((k, \delta)\) isolated nbhd of \( v \)
    - Ignoring any node which is in \( P[w] \) for \( w \) with smaller rank
    - If find it, \( P[v] \leftarrow \) this nbhd
    - Else \( P[v] \leftarrow \epsilon v^3 \)

**Query complexity:**

- \( O(k) \) nodes within distance \( 2k \)
- \( 2^{O(k)} \) using [NO] analysis
  - \( \Theta(k \times \frac{\epsilon^3}{\text{big constant}}) \)