

Lecture 5

Distributed algorithms

vs.

sublinear algorithms

(on sparse graphs)



max degree $\leq \Delta$
constant!

Today:

a way of designing lots of
sublinear algorithms for sparse
(ave deg $O(1)$) graphs

idea: use known fast distributed algs
+
reduction

Vertex Cover: $V' \subseteq V$ is "Vertex cover" (VC)

if $\forall (u,v) \in E$, either $u \in V'$ or $v \in V'$

VC question: What is min size of VC?

note: • in $\deg \leq \Delta$ graph, $|VC| \geq \frac{m}{\Delta} \geq \frac{n \cdot \Delta_{\text{ave}} \cdot \frac{1}{2}}{\Delta_{\text{max}}}$
since each node covers $\leq \Delta$ edges

• in general VC is NP-complete

• there is poly time 2-approx via matching

Question: Can you approx V.C. in Sublinear time?

multiplicative? no! can't distinguish graph with
0 edges from graph with
1 edge in sublinear time

additive? hard since NP-complete
need some mult error (at least 1.36
maybe factor of 2)

Combination mult & add error?

def $\hat{y}$ is (α, ϵ) -approximation of soln value y
for minimization problem if

$$y \leq \hat{y} \leq \alpha \cdot y + \epsilon$$

(analogous defn for maximization problem)

(in our case of $\Delta = O(1)$, additive error can
be "hidden" in multiplicative error)

Background on LOCAL distributed Algorithms:

↑ well studied model

- Network

- processors
 - links
- $\sum$ max degree Δ

- Communication round

- nodes perform computation on
(input bits, history of received msgs,
random bits)
- nodes send msgs to nbrs
- nodes receive msgs from nbrs

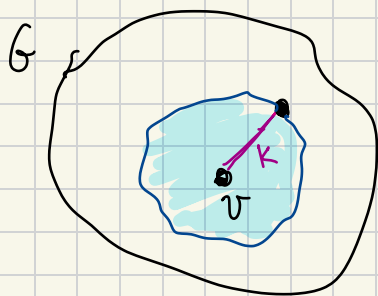
- Vertex cover on a distributed network:

- network graph = input graph
- at end, each node knows if it is in VC
(doesn't necessarily know about other nodes
or total size)

LOCAL Distributed vs, Sublinear time:

main insight:

In k -round LOCAL algorithm,
output of node v can only
depend on nodes of distance $\leq k$
from v . At most Δ^k of these!



$\Rightarrow$ Can sequentially simulate v 's view
of distributed computation in $\leq \Delta^k$ queries
& at end know if v in or out of VC

Comment: if LOCAL algo is randomized, v needs to know
random bits of all $\leq \Delta^k$ nbrs
 $\uparrow$ must be consistent

So fast distributed LOCAL for VC

$\Rightarrow$ fast sequential "oracle" for VC
sublinear query? what about sublinear time?

Is there a fast distributed LOCAL alg for VC?

Yes, will see more soon

What do we do with this oracle?

use to estimate size of VC

via sampling.

Estimating size of VC

Parnas-Ron Framework:

Sample nodes of graph $V_1 \dots V_r$

for each v_i

simulate distributed LOCAL algo to see if $v_i \in VC$

Output $\frac{\# v_i \text{'s in } VC}{r} \cdot n$

runtime: $O(r \cdot \Delta^{k+1}) \approx O(\frac{1}{\epsilon^2} \cdot \Delta^{k+1})$

$k = \# \text{ rounds of distr. algo.}$
 $\Delta = \text{max degree}$

Proof of correctness: Chernoff-Hoeffding bnds

A fast distributed algorithm for VC

idea: primal - dual based approx algo
 ↑ ↑
 matching V.C.

Relate V.C. to dual problem of fractional matching!

What is a fractional matching?

- assign weights x_e in $[0,1]$ to edges e

st. $\forall v \sum_{e \ni v} x_e \leq 1$

edges containing v
as end pt

- value of fract matching = $\sum_e x_e$

in integral matching,
 $e \in \{0,1\}$

+ $\sum_{e \ni v} x_e$ is

either 0 or 1
unmatched or matched

Algorithm

initially sum around each node ≤ 1

- Init $x_e = \frac{1}{\Delta} \quad \forall \text{ edge } e$

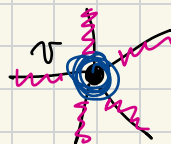
← start with trivial soln to matching (primal)

- For $i=1$ to $\log_{1+\delta} \Delta$

- for each node v s.t. $\sum_{e \ni v} x_e \geq \frac{1}{1+\delta}$,

- add v to V_L

- freeze $x_e \quad \forall e \ni v$
(v sends "frozen" msg to all nbors)



- for each unfrozen edge e , set $x_e \leftarrow (1+\delta)x_e$

(At end any remaining v are not in V_L .)

↑
gradually increase until become maximal

#rounds: $1 + \log_{1+\delta} \Delta$

time to simulate: $\Delta^{\log_{1+\delta} \Delta}$

Why is it a V.C.?

if e frozen, then ≥ 1 endpt in V.C., else

$$\text{if } e \text{ survives until end, } X_e > (1+r)^{\log_{1+r} \Delta} \cdot \frac{1}{\Delta} = \frac{\Delta}{\Delta} > 1$$

$\Rightarrow$ endpts would join V.C.
 $\Rightarrow$ no e survives

$\Rightarrow$ at least one endpt in V.C. ~~■~~

Why is it a small V.C.?

Note: our algorithm finds a fractional matching

why? if v doesn't freeze edges,

$$\sum_{e \ni v} x_e < \frac{1}{1+\epsilon}$$

so in next round $\sum_{e \ni v} x_e \leq \left(\frac{1}{1+\epsilon}\right) \cdot (1+\epsilon) = 1$

$\Rightarrow$ gets frozen before getting ~~bigger~~ than 1.

wt of our fractional matching $= \sum_e x_e$

plan: show (1) ^(any) fractional matching value $\leq$ min V.C. ↙ integral

(2) ^(our) Output V.C. $\leq 2(1+\epsilon) \cdot$ ^(our) fractional matching value

$\Rightarrow$ Output V.C. $\leq 2 \cdot (1+\epsilon) \cdot \text{min V.C.}$

Any fractional matching value $\leq \min \text{V.C.}$:
in particular, max + the one found in our algorithm

recall: integral max matching $\leq \min \text{V.C.}$
since each edge in matching contributes ≥ 1 node to any V.C.

Given valid fractional matching with value $\sum_e x_e$
+ optimal V.C. V_{opt}
 $\forall e$, assign x_e to endpt in $V_{\text{opt}} \Rightarrow$ *if both in V_{opt} , choose arbitrarily*
total wt assigned to nodes = $\sum_e x_e$
each node in V_{opt} gets ≤ 1
since comes from fract. matching

$$\Rightarrow |V_{\text{opt}}| \geq \frac{\text{total wt}}{\text{max wt per node}} \geq \frac{\sum_e x_e}{1} = \sum_e x_e$$



Output V.C. $\leq 2(1+\delta)$ ^(our) Fractional matching value

Given output V.C. $\hat{V}$

for every $u \in \hat{V}$, assign value 1
(so total value = $|\hat{V}|$)

Split u 's value among neighboring edges
proportional to X_e : $\leftarrow$ from algorithm

each edge e' gets $\frac{X_{e'}}{\sum_{e \in u} X_e} \leq \frac{X_{e'}}{\frac{1}{1+\delta}} = (1+\delta)X_{e'}$ from u

can also get at most $(1+\delta)X_{e'}$ from other
endpt

total assigned to $e' \leq 2(1+\delta)X_{e'}$

total value = $|\hat{V}| \leq 2(1+\delta) \sum_e X_e$

wt of our
fract.
matching

