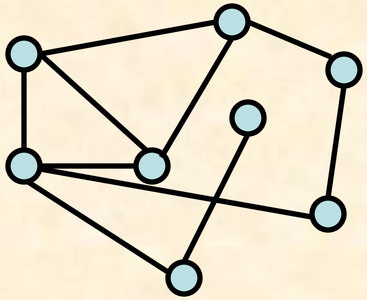
Sublinear Algorithms for Approximating Graph Parameters

Dana Ron
Tel-Aviv University

TCE Conference, May 2013

Graph Parameters

A Graph Parameter: a function σ that is defined on a graph G (undirected / directed, unweighted / weighted).

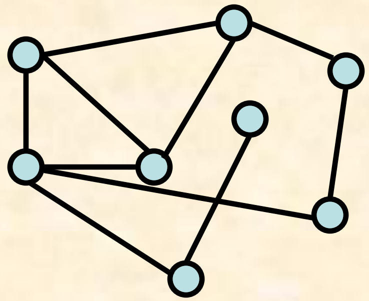


For example:

- Average degree
- Number of subgraphs H in G
- Number of connected components
- Minimum size of a vertex cover
- Maximum size of a matching
- Number of edges that should be added to make graph k-connected (distance to k-connectivity)
- Minimum weight of a spanning tree

Computing/Approximating Graph Parameters Efficiently

For all parameters described in the previous slide, have efficient, i.e., polynomial-time algorithms for computing the parameter (possibly approximately). For some even linear-time.



However, in some cases, when inputs are very large, we might want even more efficient algorithms: sublinear-time algorithms.

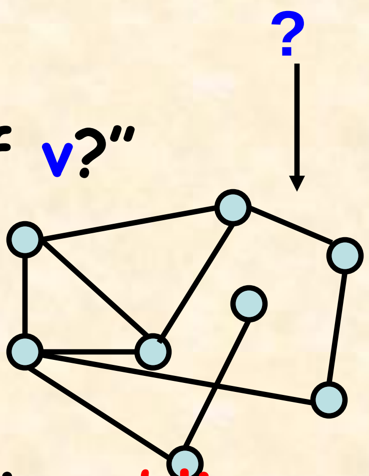
Such algorithms do not even read the entire input, are randomized, and provide an approximate answer (with high success probability).

Sublinear Approximation on Graphs

Algorithm is given query access to G.
Types of queries that consider:

- Neighbor queries – "who is i^{th} neighbor of v?"
- Degree queries – "what is $deg(v)$?"
- Vertex-pair queries – "is there an edge btwn u and v?"

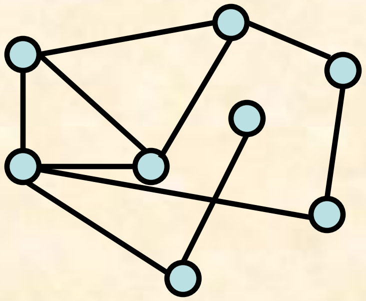
+ weight of edge



After performing number of queries that is sublinear in size of G, should output good approximation σ' of $\sigma(G)$, with high success probability.
Types of approximation that consider:

- $\sigma(G) \le \sigma' \le (1+\varepsilon)\cdot\sigma(G)$ (for given ε : a $(1+\varepsilon)$-approx.)
- $\sigma(G) \le \sigma' \le \alpha\cdot\sigma(G)$ (for fixed α : an α-approx.)
- $\sigma(G) \le \sigma' \le \alpha\cdot\sigma(G) + \varepsilon n$ (for fixed α and given ε where n is size of range of $\sigma(G)$: an (α, ε)-approx.)

Survey Results in 4 Parts



I. Average degree and number of subgraphs

II. Minimum weight spanning tree

III. Minimum vertex cover (and maximum matching)

IV. Distance to having a property (e.g. k-connectivity)

Part I: Average Degree

Let $d_{avg} = d_{avg}(G)$ denote average degree in G, $d_{avg} \geq 1$

Observe: approximating average of general function with range $\{0,...,n-1\}$ (degrees range) requires $\Omega(n)$ queries, so must exploit non-generality of degrees

Can obtain $(2+\varepsilon)$-approximation of d_{avg} by performing $O(n^{1/2}/\varepsilon)$ degree queries [Feige].

Going below 2: $\Omega(n)$ queries [Feige].

With degree and neighbor queries, can obtain $(1+\varepsilon)$-approximation by performing $\tilde{O}(n^{1/2}\ poly(1/\varepsilon))$ queries [Goldreich,R].

Comment1: In both cases, can replace $n^{1/2}$ with $(n/d_{avg})^{1/2}$

Comment2: In both cases, results are tight (in terms of dependence on n/d_{avg}).

Part I: Average Degree

Ingredient 1: Consider partition of all graph vertices into $r=O((\log n)/\varepsilon)$ buckets: In bucket B_i vertices v s.t.

$$(1+\beta)^{i-1} < \deg(v) \leq (1+\beta)^i \quad (\beta = \varepsilon/8)$$

Suppose can obtain for each i estimate $b_i = |B_i| \cdot (1 \pm \beta)$

$$(1/n) \cdot \Sigma_i\, b_i \cdot (1+\beta)^i = (1 \pm \varepsilon) \cdot d_{avg} \qquad (*)$$

How to obtain b_i? By sampling (and applying [Chernoff]).

Difficulty: if B_i is small ($<< n^{1/2}$) then necessary sample is too large ($(|B_i|/n)^{-1} >> n^{1/2}$).

Ingredient 2: ignore small B_i's. Take sum in $(*)$ only over large buckets ($|B_i| > (\varepsilon n)^{1/2}/2r$).

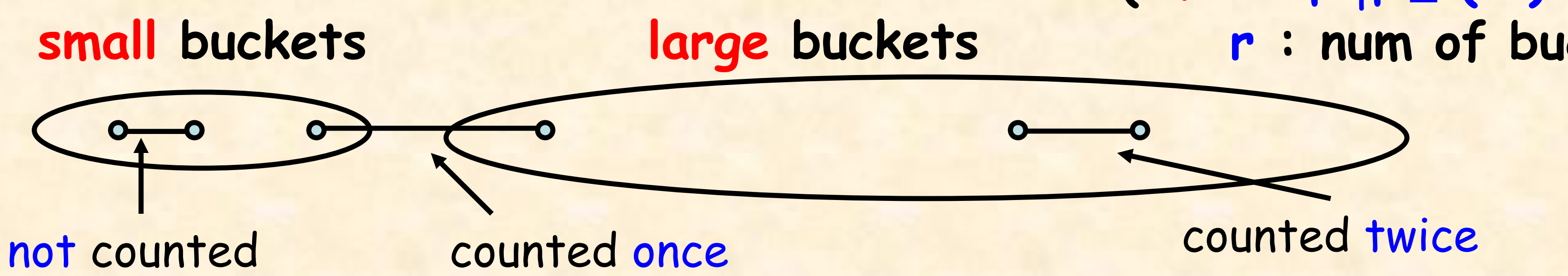
Claim: $$(1/n) \cdot \Sigma_{\text{large } i}\, b_i \cdot (1+\beta)^i \geq d_{avg}/(2+\varepsilon) \qquad (**)$$

Part I: Average Degree

Claim: $(1/n)\cdot\Sigma_{\text{large } i}\ b_i\cdot(1+\beta)^i \geq d_{avg}/(2+\varepsilon)$ (**)

Sum of degrees = 2 · num of edges

(small: $|B_i| \leq (\varepsilon n)^{1/2}/2r$, r : num of buckets)

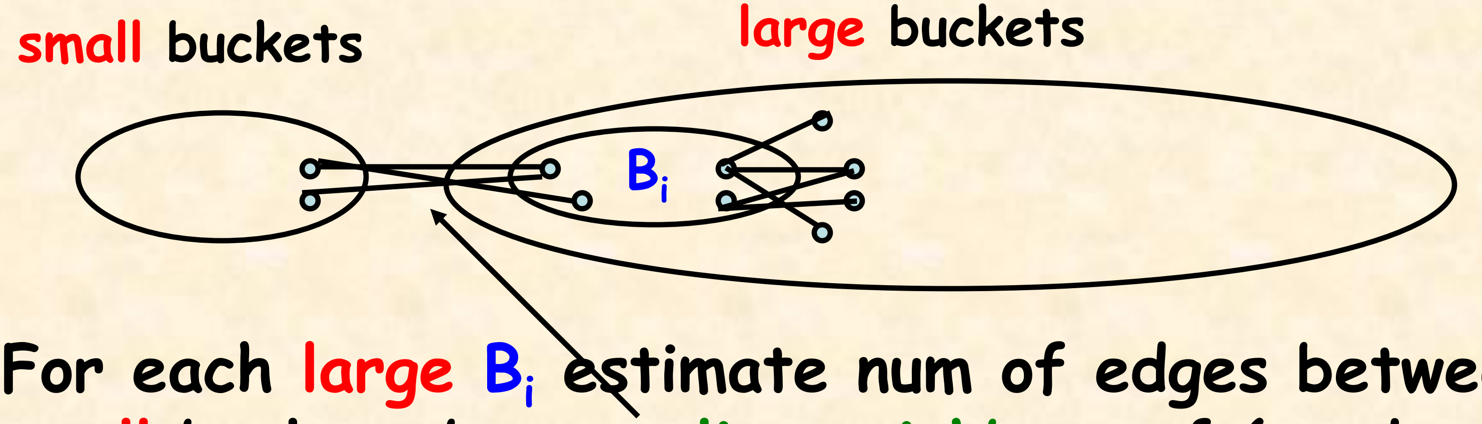


Using (**) get $(2+\varepsilon)$-approximation with $\tilde{O}(n^{1/2}/\varepsilon^2)$ degree queries

Ingredient 3: Estimate num of edges counted once and compensate for them.

Part I: Average Degree

Ingredient 3: Estimate num of edges counted once and compensate for them.



For each large B_i estimate num of edges between B_i and small buckets by sampling neighbors of (random) vertices in B_i.

By adding this estimate e_i to (**) get $(1+\varepsilon)$-approx.

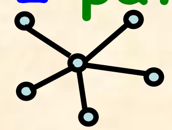
$(1/n)\cdot\Sigma_{large\ i}\ b_i\cdot(1+\beta)^i$ (**)

$$(1/n)\cdot\Sigma_{large\ i}\ (b_i\cdot(1+\beta)^i + e_i)$$

Part I(b): Number of stars subgraphs

Approximating avg. degree same as approximating num of edges. What about other subgraphs? (Also known as counting network motifs.)

[Gonen,R,Shavitt] considered length-2 paths, and more generally, s-stars.



(avg deg + 2-stars gives variance, larger s – higher moments)

Let $N_s=N_s(G)$ denote num of s-stars. Give $(1+\varepsilon)$-approx algorithm with query complexity (degree+neighbors):

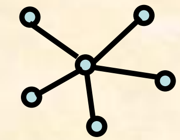
$$O\left(\frac{n}{N_s^{1/(s+1)}}+\min\left\{n^{1-1/s},\frac{n^{s-1/s}}{N_s^{1-1/s}}\right\}\right)poly(\log n,1/\varepsilon)$$

Show that this upper bound is tight.

Part I(b): Number of stars subgraphs

$$O\left(\frac{n}{N_s^{1/(s+1)}} + \min\left\{n^{1-1/s}, \frac{n^{s-1/s}}{N_s^{1-1/s}}\right\}\right) poly(\log n, 1/\varepsilon)$$

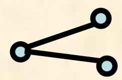
$N_s \le n^{1+1/s}$: $O(n/(N_s)^{1/(1+s)})$



$n^{1+1/s} \le N_s \le n^s$: $O(n^{1-1/s})$

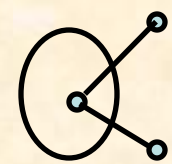
$N_s > n^s$: $O(n^{s-1/s}/(N_s)^{1-1/s}) = O((n^{s+1}/N_s)^{1-1/s})$

Example: $s=3$. (a) $N_s = n$: $O(n^{3/4})$; (b) $N_s = n^2$: $O(n^{2/3})$; (c) $N_s = n^4$: $O(1)$



Idea of algorithm for $s=2$: Also partition into buckets.

Can estimate num of 2-stars with centers in large buckets.



Also estimate num of 2-stars with centers in ("significant") small buckets and at least one endpoint in large bucket by estimating num of edges between pairs of buckets.

$$b_i \cdot \binom{(1+\beta)^i}{2}$$

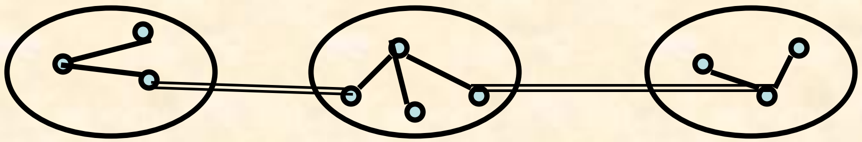
Part II: MST

Consider graphs with degree bound d and weights in $\{1,...,W\}$.

[Chazelle, Rubinfeld, Trevisan] give $(1+\varepsilon)$-approximation alg using $\tilde{O}(d \cdot W/\varepsilon^2)$ neighbor queries.

Result is tight and extends to $d=d_{avg}$ and weights in $[1,W]$.

Suppose first: $W=2$ (i.e., weights either 1 or 2)
E^1 = edges with weight 1, $G^1=(V,E^1)$, c^1 = num of connected components in G^1.

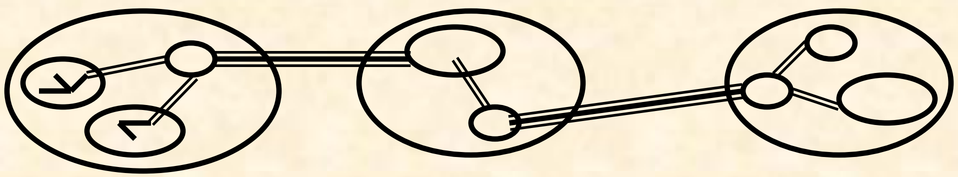


Weight of MST: $2 \cdot (c^1-1) + 1 \cdot (n-1-(c^1-1)) = n-2+c^1$

Estimate MST weight by estimating c^1

Part II: MST

More generally (weights in $\{1,..,W\}$)
E^i = edges with weight $\leq i$, $G^i=(V,E^i)$, c^i = num of connected components (cc's) in G^i.

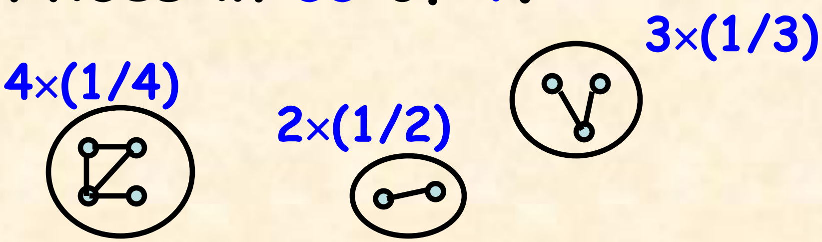


Weight of MST: $n - W + \Sigma_{i=1..W-1}\ c^i$

Estimate MST weight by estimating $c^1,\ldots,c^{W-1}$.

Idea for estimating num of cc's in graph H ($c(H)$):
For vertex v, n_v = num of vertices in cc of v.

Then: $c(H) = \Sigma_v(1/n_v)$



Part II: MST

$c(H) = \Sigma_v(1/n_v)$ (n_v = num of vertices in cc of v)

Can estimate c(H) by sampling vertices v and finding n_v for each (using BFS).

Difficulty: if n_v is large, then "expensive"

Let $S = \{v : n_v \leq 1/\beta\}$.

$$\Sigma_{v \in S}(1/n_v) \geq c(H) - n/(1/\beta) = c(H) - \beta n$$

Alg for estimating c(H) selects $\Theta(1/\beta^2)$ vertices, runs BFS on each selected v until finds n_v or determines that $n_v > 1/\beta$ (i.e. $v \notin S$). Uses $\Sigma(1/n_v)$ for sampled vertices in S to estimate c(H). Complexity: $O(d/\beta^3)$

Alg for estimating MST can run above alg on each G^i with $\beta=\varepsilon/(2W)$ (so that when sum estimates of c^i over $i=1,\ldots,W$ get desired approximation).

Comment: [Chazelle,Rubinfeld,Trevisan] get better complexity (total of $\tilde{O}(d \cdot W/\varepsilon^2)$) by more refined alg

Part III: Min VC

Initially considered in [Parnas,R].

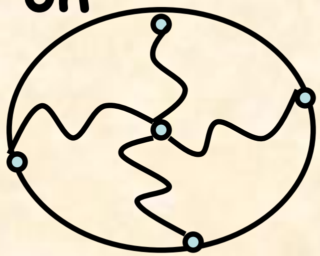
First basic idea: Suppose had an oracle that for given vertex v says if $v \in VC$ for some fixed VC that is at most factor α larger than min VC.

Can get (α,ε)-approximation by sampling $\Theta(1/\varepsilon^2)$ vertices and querying oracle.

$vc(G) \leq vc' \leq \alpha\, vc(G) + \varepsilon n$

Second idea: Can use distributed algorithms to implement oracle. In dist algs on graphs have processor on each vertex. In each round send messages to all neighbors. At end, each processor knows answer (e.g. is vertex in cover)

If dist. alg. works in k rounds of communication, then oracle, when called on v, will emulate dist. alg. on k-distance-neighborhood of v



Query complexity of each oracle call: $O(d^k)$

Part III: Min VC

By applying dist. alg. of [Kuhn, Moscibroda, Wattenhofer] get (c,ε)-approx. ($c>2$) with complexity $d^{O(\log d)}/\varepsilon^2$, and $(2,\varepsilon)$-approx. with complexity $d^{O(d)\mathrm{poly}(1/\varepsilon)}$.

Comment 1: Can replace max deg d with d_{avg}/ε [PR]

Comment 2: Going below 2 : $\Omega(n^{1/2})$ queries (Trevisan)
7/6: $\Omega(n)$ [Bogdanov, Obata, Trevisan]

Comment 3: Any (c,ε)-approximation: $\Omega(d_{avg})$ queries [PR]

Sequence of improvements for $(2,\varepsilon)$-approx

[Marko, R]: $d^{O(\log(d/\varepsilon))}$ - using dist. alg. similar to max ind. set alg of [Luby])

[Nguyen, Onak]: $2^{O(d)}/\varepsilon^2$ – emulate classic greedy algorithm (maximal matching) [Gavril], [Yanakakis]

[Yoshida, Yamamoto, Ito]: $O(d^4/\varepsilon^2)$– better emulation

[Onak, R, Rosen, Rubinfeld]: $\tilde{O}(d_{avg}\,\mathrm{poly}(1/\varepsilon))$

Part III(b): Maximum Matching and more

[Nguyen,Onak] give $(1,\varepsilon)$-approx for max match with complexity $2^{d*O(1/\varepsilon)}$, improved [Yoshida,Yamamoto,Ito] to $d^{6/\varepsilon*2}(1/\varepsilon)^{O(1/\varepsilon)}$
Recursive application of oracles using augmenting paths.

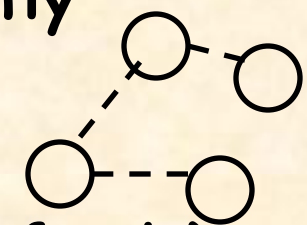
[Hassidim,Kelner,Nguyen,Onak],[Elek] give $(1,\varepsilon)$-approx algs on restricted graphs (e.g., planar)

Can get $(O(\log d),\varepsilon)$-approx for min dominating set with complexity $d^{O(\log d)}/\varepsilon^2$ using [Kuhn,Moscibroda,Wattenhofer]

Part IV: Approximating Distance to P

For graph property P, estimate fraction of edges that should be added/removed to obtain P (fraction with respect to (ub on) num of edges m). Assume $m=\Omega(n)$.

Study of distance approximation first explicitly introduced in [Parnas,R,Rubinfeld].



Note: Already discussed alg for distance to connectivity in sparse graphs (estimate num of cc's)

For dense graphs where $m=\Omega(n^2)$ and perform vertex-pair queries, some known testing results directly give dist. approx. results: e.g., ρ-cut (having a cut of size at least ρn^2): $(1,\varepsilon)$-approx using $poly(1/\varepsilon)$ queries ($exp(poly(1/\varepsilon))$ time) – equiv to approx Max-Cut.

[Fischer,Newman]: all testable properties (comp. independent of n) have dist. approx. algs. Direct Analysis for monotone properties [Alon,Shapira,Sudakov]

Part IV: Approximating Distance to P

Dist. app. for sparse graphs studied in [Marko,R]

distance w.r.t, $d \cdot n$

Property	Model	α	Complexity
k-Edge-Connectivity	sparse	1	$poly(k/(\varepsilon\, d_{avg}))$
Triangle-Freeness	bounded-degree	3	$d^{O(\log(d/\varepsilon))}$
Eulerian	sparse	1	$O(1/(\varepsilon\, d_{avg})^4)$
Cycle-Freeness	bounded-degree	1	$O(1/\varepsilon^3)$

$\Omega(n^{1/2})$ for sparse model

Extends to subgraph-free

cannot get sublin with $\alpha=1$

[Hassidim,Kelner,Nguyen,Onak] give $(1,\varepsilon)$-approx for restricted graphs: e.g. dist. to 3-col in planar graphs.

Summary

Presented sublinear approximation algorithms for various graph parameters:

I. Average degree and number of subgraphs

II. Minimum weight spanning tree

III. Minimum vertex cover and maximum matching

IV. Distance to having a property (e.g. k-connectivity)

Thanks