Sublinear time algorithms I

Ronitt Rubinfeld
MIT

FODSI Summer School August 2022

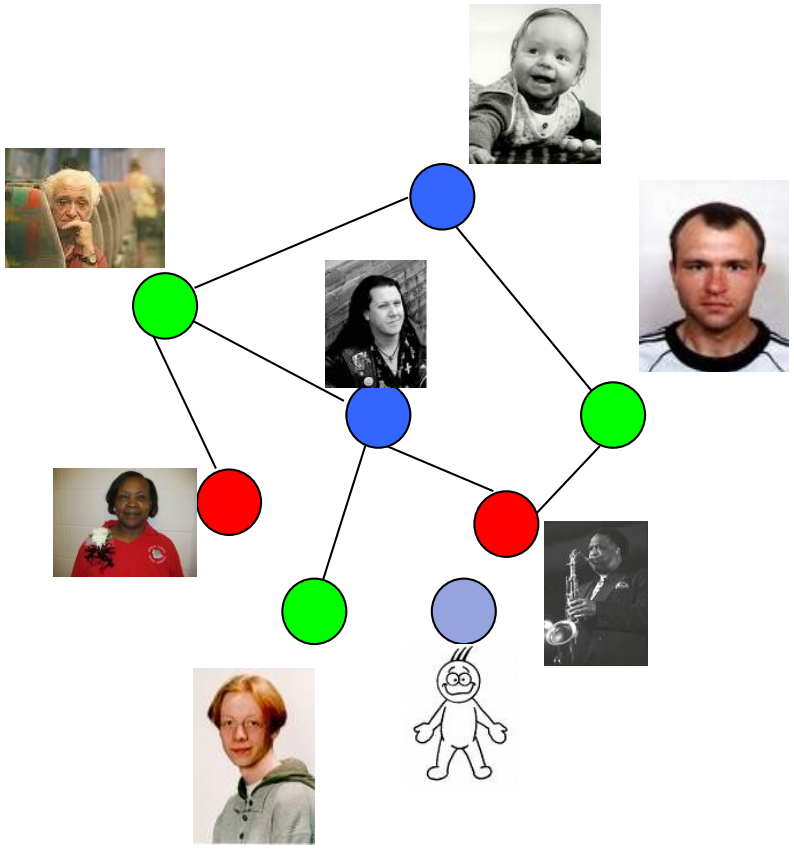
Big data?

Really Big data

Impossible to access all of it

Social network

- each “node’’ is a person
- “edge’’ between people that know each other



Small world property

"Six degrees of separation"

In our language:

diameter of the world population is 6

Does earth have the small world property?

- How can we know?
  - data collection problem is immense
  - unknown groups of people found on earth
  - births/deaths

- Stanley Milgram's 1963 experiment?

The Gold Standard

- linear time algorithms
  - Inadequate…

What can we hope to do without viewing most of the data?

- Can't answer "for all" or "there exists" and other "exactly" type statements:
  - are *all* individuals connected by at most 6 degrees of separation?
  - *exactly* how many individuals on earth are left-handed?

- Maybe can answer?
  - is there a *large* group of individuals connected by at most 6 degrees of separation?
  - is the *average* pairwise distances of a graph roughly 6?
  - *approximately* how many individuals on earth are left-handed?

What can we hope to do without viewing most of the data?

- Must compromise:
  - for most interesting problems: algorithm must give *approximate* answer
- we know we can answer *some* questions...
  - e.g., sampling to approximate average, median values

Plan

- Today:
  - Diameter of point set
  - Estimate the degree of a graph
  - Estimate the number of connected components of a graph
  - Estimate Minimum Spanning Tree weight

- Tomorrow:
  - Sublinear algorithms from distributed algorithms
  - Sublinear algorithms from greedy algorithms
  - Property testing

First:

- A very simple example –
  - Deterministic
  - Approximate answer
  - And (of course)…. sub-linear time!

Approximate the diameter of a point set

- Given: m points, described by a distance matrix D, s.t.
  - D_{ij} is the distance from i to j.
  - D satisfies triangle inequality and symmetry.

  (note: input size $n = m^2$)
- Let i, j be indices that maximize D_{ij} then D_{ij} is the *diameter.*
- Output: k,l such that $D_{kl} \geq D_{ij}/2$

2-multiplicative approximation

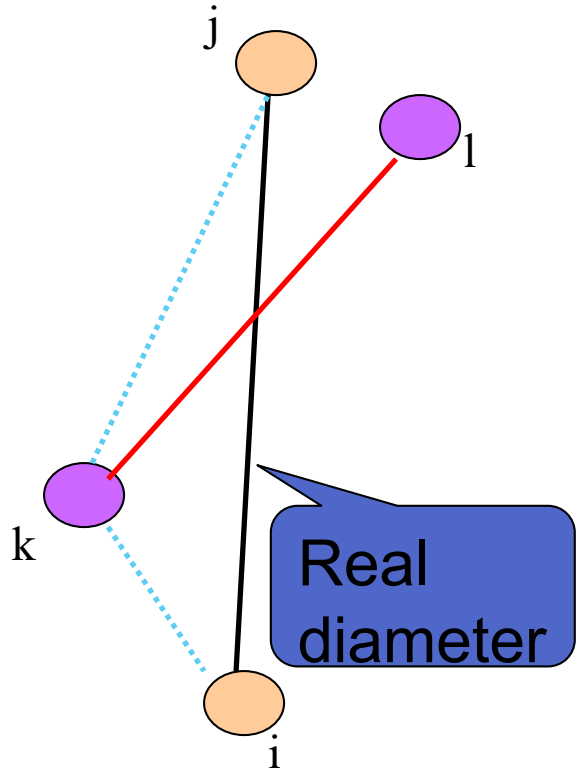
Algorithm

- Algorithm:
  - Pick *k* arbitrarily
  - Pick *l* to maximize D_{kl}
  - Output D_{kl}
- Running time? $O(m) = O(n^{1/2})$
- Why does it work?

$D_{ij} \le D_{ik} + D_{kj}$ (triangle inequality)

$\le D_{kl} + D_{kl}$ (choice of *l* + symmetry of *D*)

$\le 2D_{kl}$ (so D_{kl} is at least diameter/2)



Average degree of a graph

[Feige][Goldreich Ron]

Average degree of a graph

- $G = (V,E)$
  - Simple (no parallel edges, self-loops)
  - $d(v)$ = degree of node v
  - $D = (1/n) \cdot \sum d(v)$ is average degree
  - Assume $\Omega(n)$ edges in graph

- Representation and access to G:
  - Adjacency list
    - For each v, store $d(v)$ followed by length $d(v)$ array of neighbors
  - Possible Queries
    - Degree queries: for any v return $d(v)$
    - Neighbor queries: for any (v,j), return j^{th} neighbor of v

Not needed today

Naïve sampling algorithm

- Algorithm:
  - Pick *s=O(??)* random nodes $v_1,\ldots,v_s$
  - Output $(1/s)\sum d(v_i)$ (average degree of sample)

- Note: only uses degree queries
- How well does it work?
  - Issue: $d(v_i)$ can be anywhere in range *[1..n]*
    - Straightforward use of sampling bounds (i.e., Chernoff/Hoeffding) require $s = \Omega(n)$ samples
    - Our hope: these are not arbitrary numbers…

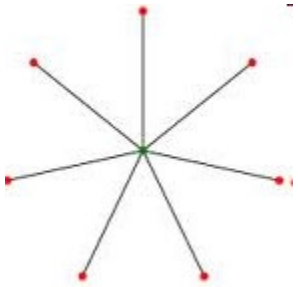
Why are degree sequences special?

- Constraints on vector of degrees:
  - Entries between *0* and *n-1*
  - Sum is *2Dn*
  - What else?

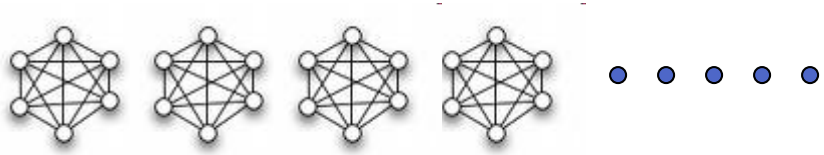
Why are degree sequences special? (continued)

- Examples:
  - Regular graph *(D,D,D,D,…D)*

  - *(n-1,0,0…0)* not possible!
  - *(n-1,1,1,1,1)* is possible

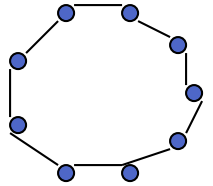


  - Collection of m different k-cliques and some isolated nodes
    (k,k,k,k,k,0,0,0,0,0)



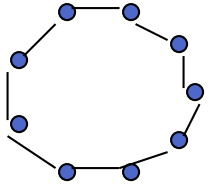
$\Omega(n^{1/2})$-Lower bound for multiplicative approximation

- *n*-cycle: average degree D is *2*



- *(n- $cn^{1/2}$)*-cycle + ($cn^{1/2}$)-clique

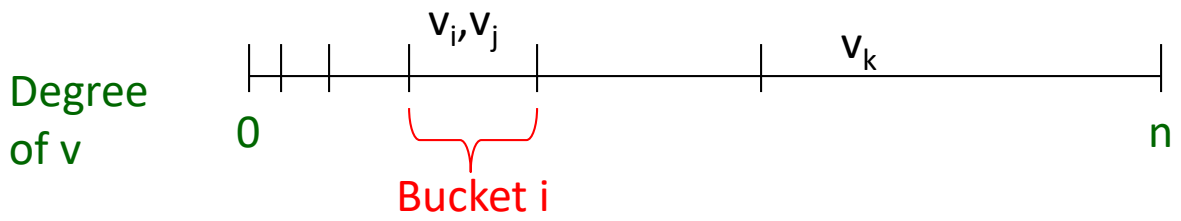
  average degree D is $\approx 2 + c^2$



Need $\Omega(n^{1/2})$ queries to find one of the clique nodes

Algorithm of [GR]: idea

- Set parameters $\beta = \frac{\epsilon}{c}, t = O(\frac{1}{\epsilon}\log n)$



- Group sampled nodes with similar degrees into same bucket
  - $B_i = \{ v \,|(1+\beta)^{i-1} \le d(v) < (1+\beta)^i\}$ for i=0…t-1
- Then total degree of nodes in bucket i is between
  $(1+\beta)^{i-1}|B_i|$ and $(1+\beta)^i|B_i|$
- Total degree of graph is between
  $\sum(1+\beta)^{i-1}|B_i|$ and $\sum(1+\beta)^i|B_i|$

Plan: estimate $|B_i|/n$ for all i

Recall:
$\sum(1+\beta)^{i-1}|B_i|$
$\leq total\ degree$
$\leq \sum(1+\beta)^{i}|B_i|$

Algorithm: First stab

- Set parameters $\beta = \frac{\epsilon}{c}, t = O(\frac{1}{\epsilon}\log n)$
- Sample nodes S
- Group sampled nodes with similar degrees into same bucket
  - $B_i = \{ v \,|(1+\beta)^{i-1} < d(v) < (1+\beta)^{i}\}$ for i=0…t-1
  - $S_i \leftarrow S \cap B_i$
- Estimate contribution to average degree from each bucket from sampled nodes that fall in it
  - $\rho_i \leftarrow \frac{|S_i|}{|S|}$ (Note that for all $i, E[\rho_i] = E[|Si| / |S|] = |Bi|/n$)
- Output $\sum(1+\beta)^{i-1}\rho_i$

Not great for buckets with small $|B_i|$

Recall:
$\sum(1+\beta)^{i-1}|B_i|$
$\leq total\ degree$
$\leq \sum(1+\beta)^{i}|B_i|$

Algorithm:

- Set parameters $\beta = \frac{\epsilon}{c}, t = O(\frac{1}{\epsilon}\log n)$
- Sample nodes S
- Group sampled nodes with similar degrees into same bucket
  - $B_i = \{ v \,|(1+\beta)^{i-1} < d(v) < (1+\beta)^{i}\}$ for i=0…t-1
  - $S_i \leftarrow S \cap B_i$
- Estimate average degree in each bucket from sampled nodes that fall in it
  - If S_i "large" (i.e., $> \left(\frac{\epsilon}{n}\right)^{\frac{1}{2}} |S|/ct$), use $\rho_i \leftarrow \frac{|S_i|}{|S|}$
  - Else use $\rho_i \leftarrow 0$
- Output $\sum(1+\beta)^{i-1}\rho_i$

$|S| = \Omega\left(\left(\frac{n}{\epsilon}\right)^{\frac{1}{2}} \cdot ct\right)$ samples guarantees good approximation for "large" i

Is it a huge undercount?

Why set $\rho_i = 0$?

- If group represents nodes with high degree, then small errors in estimating size of group size could cause major errors in estimating average degree

How many samples?

enough to get good estimate of size of buckets with $\Omega((\epsilon n)^{\frac{1}{2}})$nodes

If $\rho_i = 0$ then bucket i has $O((\epsilon n)^{\frac{1}{2}})$ nodes

Undercounting from "small" buckets

- Three types of edges:
  1. "large-large": Both endpoints in large buckets — counted twice
  2. "large-small" One endpoint in large bucket, one in small — counted once
  3. "small-small" Both endpoints in small buckets — never counted

- How many small-small edges?
  - Small buckets don't have lots of nodes:
    - Chernoff says that if i is such that $|B_i|/n$ is big – i.e., $\geq \left(\frac{\epsilon}{n}\right)^{\frac{1}{2}} /(c't)$

      then whp number of samples $|S_i| > \left(\frac{\epsilon}{n}\right)^{\frac{1}{2}} |S|/\ c\ t$ ("large")
    - So, whp, all small-small edges between buckets with $< \left(\frac{\epsilon}{n}\right)^{\frac{1}{2}} cn = O((\epsilon n)^{\frac{1}{2}})$nodes
  - So, at most $O(\epsilon n)$such edges

    affects average degree estimate by at most ϵ —multiplicative factor
- "large-small" undercounts by factor 2

  Yields 2+ϵ multiplicative approximation OVERALL

Can we do better?

- YES!

- Idea: estimate fraction of “large-small” edges and correct for them
  - If could pick a random edge, can use standard sampling (i.e., Chernoff) (“almost”-random edge suffices)

see [Goldreich Ron] [Eden Ron Seshadhri]

Estimating the number of connected components

Approximating number of connected components: [Chazelle R. Trevisan]

- Given input graph with
  - max degree *d*
  - adjacency list representation
- outputs additive approximation to within ϵn of the number of connected components in time $O\left(d\ \epsilon^2 \log\left(\frac{1}{\epsilon}\right)\right)$
- Can show $\Omega(d\ \epsilon^2)$ time is required

Approximating # of connected components

- Let c = number of components
- For every vertex u, define
  n_u := 1 / size of component of u
  - for any connected component $A \subseteq V$,
    $\sum_{u \in A} n_u = 1$
  - so $\sum_u n_u = c$

Main idea

- Estimate sum of approximations of n_u's via sampling

- To estimate $n_u \equiv 1$ / size of component of u quickly:
  - If size of component is big, then n_u is small so easy to estimate (inspired by property tester for connectivity [Goldreich Ron])
  - Suffices to compute n_u exactly only for small components

Some more details:

Estimating $n_u \equiv 1$ / size of u's component:

- let $\tilde{n}_u := \max\{n_u, \varepsilon/2\}$
  - When size of u's component is $< 2/\varepsilon$, $\tilde{n}_u = n_u$
  - Else $\tilde{n}_u = \varepsilon/2$

- $|n_u - \tilde{n}_u| < \varepsilon/2$ so $c = \sum_u n_u = \sum_u \tilde{n}_u \pm \varepsilon n / 2$

- can compute $\tilde{n}_u$ quickly
  - in time $O(1/\varepsilon)$ with BFS (will need $O(d/\varepsilon)$ for this later)

Not quite optimal algorithm:

CC-APPROX(ε):

Repeat $O(1/\varepsilon^3)$ times

pick a random vertex *v*

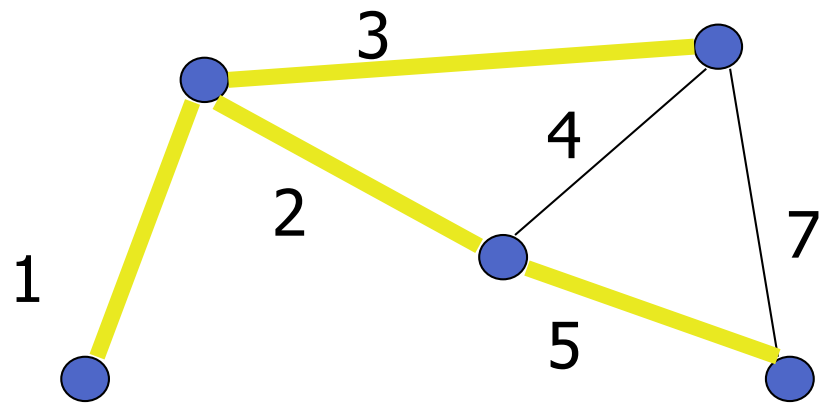
compute $\tilde{n}_v$ via BFS from *v*, stopping after at most $2/\varepsilon$ new nodes

return (average of the values $\tilde{n}_v$) • *n*

Run time: $O(d/\varepsilon^4)$

Minimum spanning tree (MST)

- What is the cheapest way to connect all the dots?



A sublinear time algorithm:

[Chazelle R. Trevisan]

Given input graph with

- weights in [1..w]
- average degree *d*
- adjacency list representation

outputs (1+ϵ)-approximation to MST in time $O(\frac{dw}{\epsilon^3}\log\frac{dw}{\epsilon})$

Remarks: (1) sublinear when *dw=o(m)*

constant when *d,w* bounded

(2) $\Omega(dw\,\epsilon^{-2})$ required

(3) case of integral weights, max degree d can be done in $O(dw\,\epsilon^{-2}\ log\ w/\epsilon)$ time

Idea behind algorithm:

- characterize MST weight in terms of number of connected components in certain subgraphs of *G*

- Use algorithm for estimating number of connected components

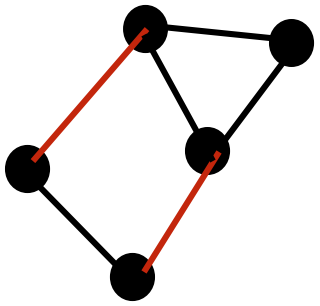
MST and connected components

Suppose all weights are 1 or 2. Then

MST weight = # weight 1 edges + 2 • # weight 2 edges

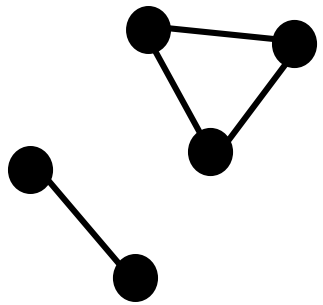
= n – 1 + # of weight 2 edges

= n – 2 + # of conn. comp. induced by weight 1 edges

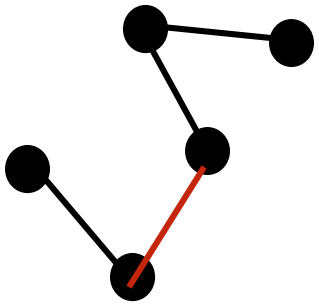


weight 1

weight 2



connected components
induced by weight 1 edges



MST

- For integer weights 1..w let

  $c^{(i)}$ = # of connected components induced by edges of weight at most i

- Then MST weight is

  $n - w + \sum_{i=1,\ldots,w-1} c^{(i)}$

- additive approximation of $c^{(i)}$'s to within $\varepsilon n/w$ gives additive approx of MST to within εn
  - Since MST > n-1, also gives multiplicative approximation of MST to within $1 \pm \varepsilon$

Improvement for MST:

This gives MST algorithm with runtime $O(dw^2 \epsilon^{-4})$

Can do better:

- $O(dw\epsilon^{-3} \log dw/\epsilon)$ algorithm

Further work:

- Euclidean MST approximation algorithm [Czumaj Ergun Fortnow Newman Magen Rubinfeld Sohler]
  - Given access to certain data structures

- Metric MST [Czumaj Sohler]
  - (1+ ε)-approximation in time $\tilde{O}(n)$

What about graph diameter?

- Distinguishing diameter k from "ϵ −far" from diameter k [Parnas Ron]
  - ϵ −far: need to add ϵm edges to decrease diameter to k
  - Runtime $poly(\frac{1}{\epsilon})$

No dependence on n

Thank you!