

What can we do in sublinear time?

0368.4612 Seminar on Sublinear Time Algorithms

Lecture 1

Ronitt Rubinfeld

Today's goal

- Motivation
- Overview of main definitions
- Examples

(Disclaimer – this is not a “complete” survey and many important references are missing, but I am always happy to receive comments)

How can we understand?



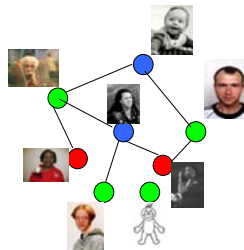
Vast data



- Impossible to access all of it
- Potentially accessible data is too enormous to be viewed by a single individual
- Once accessed, data can change

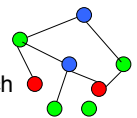
Small world phenomenon

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



Small world property

- “connected” if every pair can reach each other
- “distance” between two people is the minimum number of edges to reach one from another
- “diameter” is the maximum distance between any pair
 - “6 degrees of separation” is equivalent to “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

The Gold Standard

- linear time algorithms:
 - for inputs encoded by n bits/words, allow cn time steps (constant c)
- Inadequate...



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

- Can't answer "for all" or "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 diameter 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 change our goals:
 - for most interesting problems: algorithm must give approximate answer
- we know we can answer *some* questions...
 - e.g., sampling to approximate average, median values

What types of approximation?

- "Classical" approximation for optimization problems:
 - output is number that is close to value of the optimal solution for given input.
 - (not enough time to construct a solution)
- Property testing for decision problems:
 - output is correct answer for given input, or at least for some other input "close" to it.
- Both types of approximations are also useful for distributions

I. Classical Approximation Problems

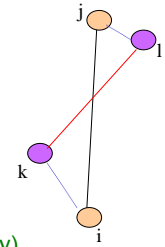
Approximate the diameter of a point set [Indyk]

- 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 $D_{kl} \geq D_{ij}/2$

Very weak approximation!

Algorithm

- Algorithm:
 - Pick k arbitrarily
 - Pick l to maximize D_{kl}
 - Output D_{kl}
- Why does it work?
 - $D_{ij} \leq D_{ik} + D_{kj}$ (triangle inequality)
 - $\leq D_{kl} + D_{kl}$ (choice of l)
 - $\leq 2D_{kl}$
- Running time? $O(m) = O(\sqrt{n})$

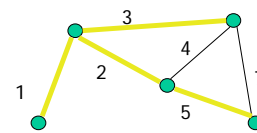


First:

- A very simple example –
 - Deterministic
 - Approximate answer

Next example: Minimum spanning tree (MST)

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



- Best known:
 - Deterministic $O(m \alpha(m))$ time [Chazelle]
 - Randomized $O(m)$ time [Karger Klein Tarjan]

A sublinear time algorithm: [Chazelle Rubinfeld Trevisan]

Given input graph with

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

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

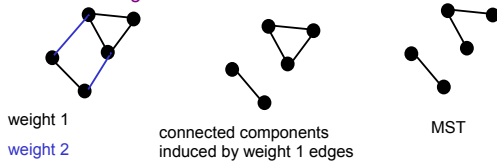
- Remarks:
- sublinear when $dw = o(m)$ constant when d, w bounded
 - we also know that $\Omega(dw\epsilon^{-2})$ required
 - case of integral weights, max degree d can be done in $O(dw\epsilon^{-2} \log w/\epsilon)$ time

Idea behind algorithm:

- characterize MST weight in terms of number of connected components in certain subgraphs of G
- show that number of connected components can be estimated quickly

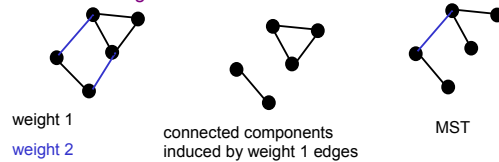
MST and connected components

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



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



- For integer weights $1..w$ let
 $c_i = \#$ of connected components induced by edges of weight at most i
- Then MST weight is
 $n - w + \sum_{i=1, \dots, w-1} c_i$
- Note that additive approximation of c_i 's to within $\epsilon n/w$ gives multiplicative approximation of MST to within $1 + \epsilon$

Approximating number of connected components:

- Given input graph with
 - max degree d
 - adjacency list representation
- outputs additive approximation to within $\epsilon_0 n$ of the number of connected components in time $O(d \epsilon_0^{-2} \log 1/\epsilon_0)$
- Can show $\Omega(d \epsilon_0^{-2})$ time is required

Approximating # of connected components

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

Main idea

- Estimate sum of approximations of n_u 's via sampling
- To estimate $n_u \equiv 1 / \text{size of component of } u$ quickly:
 - If size of component is big, then n_u is small so easy to estimate (similar to property tester for connectivity [Goldreich Ron])
 - Suffices to compute n_u exactly only for small components

Some more details:

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

- let $\tilde{n}_u := \max \{n_u, \varepsilon_0/2\}$
 - When size of u 's component is $< 2/\varepsilon_0$, $\tilde{n}_u = n_u$
 - Else $\tilde{n}_u = \varepsilon_0/2$
- $|n_u - \tilde{n}_u| < \varepsilon_0/2$ so $c = \sum_u \tilde{n}_u \pm \varepsilon_0 n / 2$
- can compute \tilde{n}_u quickly
 - in time $O(d/\varepsilon_0)$ with BFS

Not quite optimal algorithm:

CC-APPROX(ε_0):

- Repeat $O(1/\varepsilon_0^2)$ times
 - pick a random vertex v
 - compute \tilde{n}_v via BFS from v , stopping after $2/\varepsilon_0$ new nodes
 - return (average of the values \tilde{n}_v) $\cdot n$

Run time: $O(d / \varepsilon_0^3)$

Improvement for MST:

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

Can do better:

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

Further work:

- Euclidean MST approximation algorithm [Czumaj Ergun Fortnow Newman Magen Rubinfeld Sohler]
 - Given access to certain data structures can do better
- Metric MST [Czumaj Sohler]
 - $(1 + \varepsilon)$ -approximation in time $\tilde{O}(n)$

Other sublinear time approximations:

- Dense problems:
 - Max cut, graph bisection, general partitioning [Goldreich Goldwasser Ron]
 - Using low rank approximations: [Frieze Kannan]
 - Max cut
 - quadratic assignment
 - conductance
 - minimum linear arrangement
 - maximum acyclic subgraph
 - Max-SNP
 - Non-boolean Constraint Satisfaction [Andersson Engebretsen] [Alon, de la Vega, Kannan, Karpinski] [Drineas Kannan Mahoney]

Yet more sublinear time approximations:

- Metric problems:
 - Furthest pair, max spanning tree, max traveling salesman, k-median... [Indyk]
 - Clustering [Indyk][Mishra Oblinger Pitt] [Alon Dar Parnas Ron]
- String problems –
 - Edit distance [Batu Ergun Kilian Magen Raskhodnikova Rubinfeld Sami]
 - Compressibility [Raskhodnikova Ron Rubinfeld Smith]
- Linear Programming – [AdVKK],[DKM]
- Matching, Vertex cover, Set cover (on sparse graphs) [Parnas Ron][Onak Nguyen]

What can we do in sublinear time?

0368.4612 Seminar on Sublinear Time Algorithms

Lecture 2

Ronitt Rubinfeld

A quick review of some aspects of probability theory:

- Remember expectation?
 $E[x] = \sum a \Pr(x = a)$
- Simple facts: Given bin with red and blue balls. Suppose red balls are at least δ fraction of total.
 - Probability that $O(1/\delta)$ draws don't see any red balls?
 $\leq (1-\delta)^{O(1/\delta)} \leq e^{-c} \leq 1/3$ (for proper choice of c)
 - Probability that $O(\log(1/\alpha)/\delta)$ draws don't see any red balls?
 $\leq (1-\delta)^{O(\log(1/\alpha)/\delta)} \leq (1/3)^{O(\log(1/\alpha))} \leq \alpha$ (for proper choice of c)

What is the probability that a random variable deviates from its expectation?

- Focus on sums/averages of n bounded variables:
 $S = \sum y_i$ where y_i in $[0, 1]$
- Note: can get similar bounds for other bounded variables

When you don't know anything about random variable y_i :

- If y_i positive, but don't know anything else, then can use *Markov's inequality* to bound $S = \sum y_i$
 $\Pr(S \geq t E[S]) \leq 1/t$
- Example:
 - toss 100 coins, Prob (number of heads ≥ 60) $\leq 5/6$
- Pros:
 - Gives some upper bound on S not being too large
 - Don't even need that y_i are bounded
 - Don't need any independence -- works even if coins taped together!
- Cons:
 - Not a great bound
 - Doesn't say anything about how likely it is that S is too small.

Bounding $S = \sum y_i$ when y_i 's are independent I

- Bounding multiplicative error
Chernoff Bounds:
 $\Pr(S > (1+\delta)E[S]) \leq e^{-\delta^2 E[S]/3}$
 $\Pr(S < (1-\delta)E[S]) \leq e^{-\delta^2 E[S]/2}$
- Example: toss 100 coins independently,
 $\Pr(\text{number of heads} \geq 60) \leq e^{-(1/5)^2(50)/2} = e^{-2.5} = 0.51$
- Pros:
 - Gives dramatically better bounds for large enough n
 - Bounds both S too big and S too small
- Cons:
 - Need sum of independent variables

Bounding $S = \sum y_i$ when y_i 's are independent II

- Bounding additive error
Hoeffding Bounds
 $\Pr(S - E[S] > \delta n) \leq e^{-2\delta^2}$
 $\Pr(S - E[S] < -\delta n) \leq e^{-2\delta^2}$
- Example: toss 100 coins independently,
 $\Pr(\text{number of heads} \geq 60) \leq e^{-2 \cdot 100 \cdot (1/10)^2} = 0.13$

One more tool:

- Chebyshev Bound

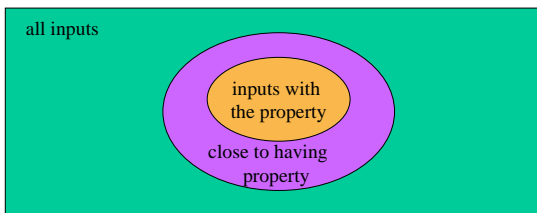
$$\Pr(|S - E[S]| > t * S.D.(S)) \leq 1/t^2$$

- Gives much better bound than Markov when can nontrivially bound *variance* of S
- Not as good as Chernoff, but doesn't require full independence

II. Property testing

Main Goal:

- Quickly distinguish inputs that *have* specific property from those that are *far* from having the property



Property Testing

- Properties of any object, e.g.,
 - Functions
 - Graphs
 - Strings
 - Matrices
 - Codewords
- Model must specify
 - representation of object and allowable queries
 - notion of close/far, e.g.,
 - number of bits that need to be changed
 - edit distance

A simple property tester

Monotonicity of a sequence

- Given: list $y_1 y_2 \dots y_n$
- Question: is the list sorted?
- Clearly requires $\Omega(n)$ time

Monotonicity of a sequence

- Given: list $y_1 y_2 \dots y_n$
- Question: can we **quickly** test if the list **close** to sorted?

What do we mean by "close"?

Definition: a list of size n is ϵ -close to sorted if can change at most ϵn values to make it sorted.

Otherwise, ϵ -far.

Requirements for property tester:

- pass sorted lists
- if list passes test, can change at most ϵ fraction of list to make it sorted

What do we mean by "quick"?

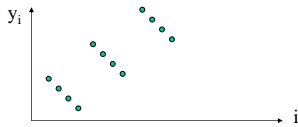
- **query complexity** measured in terms of list size n
- Our goal (if possible):
 - **sublinear** in n

Monotonicity of a sequence

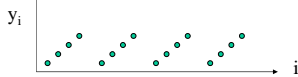
- Given: list $y_1 y_2 \dots y_n$
- Question: can we **quickly** test if the list **close** to sorted?
 - i.e., (1) pass sorted lists and (2) if passes test, can change at most ϵ fraction of list to make it sorted
- Can test in $O(1/\epsilon \log n)$ time
 - [Ergun, Kannan, Kumar, Rubinfeld, Viswanathan]
 - **best possible** [EKKR] + [Fischer]

Failed attempts:

- Pick random $i < j$ and test that $y_i \leq y_j$



- Pick random i and test that $y_i \leq y_{i+1}$



The test:

- Wlog, assume $y_1 y_2 \dots y_n$ distinct
- Define index i to be **good** if binary search for y_i successful
 - can determine goodness in $O(\log n)$ time
- $O(1/\epsilon \log n)$ time property test:
 - pick $O(1/\epsilon)$ i 's and pass if they are all good
- **Correctness:**
 - If list is sorted, then all i 's are good
 - If at least $(1-\epsilon)n$ i 's are good:
 - Main observation: good elements form increasing sequence
 - Thus list is close to sorted

Property Tester Definition:

Given input x

- If x has the property, tester passes with probability $> 2/3$
- If x is ϵ -far from having the property, tester fails with probability $> 2/3$

Comments:

- If x doesn't have property, but ϵ -close, then either output is ok
- repeat $O(\log 1/\beta)$ times and take majority for confidence $1-\beta$

History of Property Tester Definition:

- A property tester [Blum Luby Rubinfeld 1990]
- A definition [Rubinfeld Sudan 1992]
- This definition, Graph property testing [Goldreich Goldwasser Ron 1996] [Goldreich Ron 1997]
- More general distance functions [Ergun Kannan Kumar Rubinfeld Viswanathan][Parnas Ron] ...
- Tolerant testing [Parnas Ron Rubinfeld 2004]

Properties of functions

- Finite domain of size n
- Input to tester is input/output table of f
- Query access
 - on input x , get $f(x)$ in one step
- What properties does f have? e.g.,
 - monotone?
 - homomorphism?
 - convex?
 - low complexity to compute?

Example: Homomorphism Property

- Example: Want to quickly test if a function over a group is homomorphism, that is

$$\forall x, y \quad f(x) + f(y) = f(x+y)$$

– Useful for

- Checking correctness of programs computing matrix, algebraic, trigonometric functions
- Probabilistically Checkable Proofs

Is the proof of the right format?

– In these cases, enough for f to be close to homomorphism

What do we mean by "close"?

Definition: f , over domain of size n , is ϵ -close to homomorphism if can change at most ϵn values to turn it into one.

Otherwise, ϵ -far.

Homomorphism Testing

- If f is homomorphism (i.e., $\forall x, y \quad f(x) + f(y) = f(x+y)$) then test should PASS with probability $> 2/3$
- If f is ϵ -far from homomorphism then test should FAIL with probability $> 2/3$
- Note: If f not homomorphism, but ϵ -close, then either output is ok

What do we mean by "quick"?

- query complexity measured in terms of domain size n
- exact algorithm:
 - obvious $O(n^2)$
 - $\Omega(n)$ required, $O(n)$ easy
- Our goal (if possible):
 - sublinear in n
 - constant independent of n ?

Testing closeness to homomorphism:

- Can we test that for randomly chosen x, y , $f(x)+f(y)=f(x+y)$?
- Yes...
 - define the rejection probability δ_f to be the fraction of x, y pairs, such that $f(x) + f(y) \neq f(x+y)$
 - sampling problem well understood: for any δ_0 (say $\delta_0 = 2/9$), can test that $\delta_f < \delta_0$ in $O(1/\delta_0)$ queries
- but so what?
 - Does small δ_f imply f is close to a homomorphism?

How big can δ_0 be?

- Trivial: if $\delta_f < \delta_0 = 1/n^2$ then f homomorphism
 - requires $O(1/\delta_0) = O(n^2)$ queries
- Is there a δ_0 , independent of domain size, which gives bounds on closeness for f satisfying $\delta_f < \delta_0$? (i.e. so that $O(1/\delta_0) = O(1)$ queries suffice?)

Nontriviality [Coppersmith]: there exist f which often pass test but are far from a homomorphism!

- $f: \mathbb{Z}_{3^k} \rightarrow \mathbb{Z}_{3^{k-1}}$
- $f(3h+d)=h, 0 \leq h < 3^k, d \in \{-1, 0, 1\}$
- f satisfies $f(x)+f(y) \neq f(x+y)$ for only $2/9$ of choices of x, y (i.e. $\delta_f = 2/9$)
- f is $2/3$ -far from a homomorphism!

Homomorphism testing

Theorem: Let G, H be finite groups.

Given $f: G \rightarrow H$ such that

for at most $\delta_f < 2/9$ of $x, y \in G$ pairs, $f(x)+f(y) \neq f(x+y)$

then f is $\delta_f/2$ -close to a homomorphism

- [Blum Luby Rubinfeld][Coppersmith][Ben-Or Coppersmith Luby Rubinfeld]

There exist other domains, e.g., $G=\mathbb{Z}_2^n, H=\mathbb{Z}_2$, where slightly better results can be achieved

- [Bellare Coppersmith Hastad Kiwi Sudan]

Proof techniques

- Algebraic relationships + probabilistic method
- Convolutions of distributions
 - distributions for which $D \circ D = D$ are uniform on subgroup of domain
 - which distributions satisfy $D \circ D \approx D$?
 - a relationship in the wrong direction:
 - distribution D : pick random x and output $f(x)$
 - $f(x)+f(y) = f(x+y)$ for most $x, y \Rightarrow DoD$ close to D
- Fourier analytic
- Graph theoretic

More generally

- Given a functional equation specifying a function family,

$$\forall x_1, \dots, x_m, G(f(x_1), \dots, f(x_m)) = 0$$

e.g., $\tan(Ax)$ is solution to $\forall x, y, f(x+y) = (f(x)+f(y))/(1-f(x)f(y))$

when can a \forall quantifier be replaced by a “for most” quantifier, while still characterizing essentially the same functions? [Rubinfeld]

- addition theorems: (trigonometric, elliptic functions)

$$\forall x, y, f(x+y) - G(f(x), f(y)) = 0$$

- D'Alembert's equation + others

$$\forall x, y, f(x+y) + f(x-y) - 2f(x)f(y) = 0$$

- NOT functional equations relating points that are linear functions of one variable

$$\forall x, f(2x) - (2\pi)^{-1/2} 2^{2x-1/2} f(x)f(x+1/2) = 0$$

Properties of functions

- Linearity and low total degree polynomials
[Blum Luby Rubinfeld] [Bellare Coppersmith Hastad Kiwi Sudan] [Rubinfeld Sudan] [Arora Safra] [Arora Lund Motwani Sudan Szegedy] [Arora Sudan] ...
- Functions definable by functional equations – trigonometric, elliptic functions [Rubinfeld]
- Groups, Fields [Ergun Kannan Kumar Rubinfeld Viswanathan]
- Monotonicity [EKKR] [Goldreich Goldwasser Lehman Ron Samorodnitsky] [Dodis Goldreich Lehman Ron Raskhodnikova Samorodnitsky] [Fischer Lehman Newman Raskhodnikova Rubinfeld Samorodnitsky]
- Convexity, submodularity [Parnas Ron Rubinfeld] [Fattal Ron]...
- Low complexity functions, Juntas
[Parnas Ron Samorodnitsky] [Fischer Kindler Ron Safra Samorodnitsky] [Diakonikolas Lee Matulef Onak Rubinfeld Servedio Wan]...

Comments:

- Testing is significantly faster than learning/interpolating the function
 - In some cases even independent of domain size
- Finite precision case – stability theory of functional equations

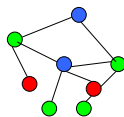
[Gemmell Lipton Rubinfeld Sudan Wigderson] [Ar Blum Codenotti Gemmell] [Ergun Kumar Rubinfeld] [Kiwi Magniez Santha]...

An open question in testing properties of functions:

- Can we characterize the properties that are testable in time independent of the domain size?

Properties of graphs [Goldreich Goldwasser Ron]

- Given graph G
- What properties does G have? e.g.,
 - bipartite? k-colorable?
 - triangle free?
 - large cut? large conductance?
 - large clique?



Dense graph model [Goldreich Goldwasser Ron]

- Representation: $n \times n$ Adjacency Matrix A
 - $A_{ij} = 1$ if edge (i,j) exists in graph
 - 0 otherwise
- Query model:
 - for any i,j can query edge A_{ij} in one step
- Distance:
 - A is ϵ -close from having property P if $\leq \epsilon n^2$ edges need to be modified

``Easily testable'' graph properties

- *Easily testable* property has tester using a number of queries independent of size of graph.
- Which are the easily testable properties?

Properties of dense graphs

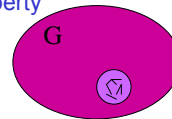
- Properties: colorability, not containing a forbidden subgraph, conductance, max cut, partition...
- *All above can be tested in constant time!!!*

Some points to keep in mind:

- *Some of these properties are NP-complete!*
- Techniques: elementary graph theory, Szemerédi's regularity lemma, low rank approximations of matrices...

Useful algorithmic technique (for dense graph model)

- Natural Algorithm: (cf. [GGR] [AFKS] ... [Goldreich Trevisan])
 - Randomly sample a set of nodes
 - Test if induced graph on sample nodes has some property



Characterization of easily testable graph properties

- Partition properties [GGR][Goldreich Trevisan]
- Logical characterization [Alon Fischer Krivelevich M. Szegedy]
- Abstract combinatorial programs [Czumaj Sohler]
- Abstract colorability properties [AFKS][Fischer]
- Properties that are ``estimable'' [Fischer Newman]
- Properties closed under vertex removal [Alon Shapira] (see also [Borgs Chayes Lovász Sós B. Szegedy Vesztergombij])
- Easily testable dense graph properties are completely characterized! [Alon Fischer Newman Shapira 2006]

Szemerédi's Regularity Lemma

- $d(C,D)$ = edge density between vertices of C and D
- Pair A,B is γ -regular if for all large enough $A' \subseteq A, B' \subseteq B$, the $|d(A',B')-d(A,B)| < \gamma$
- Equipartition V_1, \dots, V_k of V is γ -regular if most ($> 1-\gamma k^2$) pairs V_i, V_j are γ -regular
- S.R.L.: For every m and γ , there exists $T(m, \gamma)$ such that every graph has a γ -regular equipartition of order k for $m < k < T(m, \gamma)$

Characterization [AFNS]

Graph property P easily testable iff knowing regular partition of G tells you if G close to having property P

An issue...

- SRL leads to algorithms with enormous dependence on ϵ
 - tower of tower of 2's
- Can one get better testers for specific properties?

A strange characterization of bipartite graphs [Alon]

- Testing H -freeness has polynomial dependence on ϵ iff H is bipartite
- Construction based on Behrend graphs (based on sets of integers with no 3-term arithmetic progressions)

Adjacency list graph model [Goldreich Ron]

- Standard representation for graphs with average degree $o(n)$
- Distance: ($d =$ average degree)
graph is ϵ -close from having property P if $\leq \epsilon dn$ edges need to be modified

Somewhat modified for general graphs ...

A question:

- Is there a characterization of the easily testable properties in the adjacency list graph property testing model?

Properties of sparse and general graphs

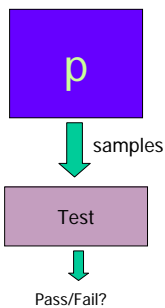
- Properties: bipartiteness, connectivity, diameter, colorability, rapid mixing, triangle free, ... [Goldreich Ron] [Parnas Ron] [Batu Fortnow Rubinfeld Smith White] [Kaufman Krivelevich Ron] [Alon Kaufman Krivelevich Ron]
- Provably different complexities in adjacency matrix vs. adjacency list models
 - cf. colorability [GGR] vs. [GR][Bogdanov Obata Trevisan]
 - Threshold-like behaviors in terms of average degree of graph [KKR] [AKKR]
- Tools more interesting algorithmically
 - random walks
 - local search

Some other combinatorial properties

- Set properties – equality, distinctness,...
- String properties – edit distance, compressibility,...
- Metric properties – metrics, clustering, convex hulls, embeddability...
- Membership in low complexity languages – regular languages, constant width branching programs, context-free languages,...
- Codes – BCH and dual-BCH codes, Generalized Reed-Muller,...

III. Testing properties of distributions

The model:



- $[n]=\{1, \dots, n\}$
- p is a black-box distribution over $[n]$, generates iid samples.
- $p_i = \text{Prob}[p \text{ outputs } i]$

Should you play the lottery? (is it uniform?)



Testing uniformity

- The goal:
 - pass uniform distribution
 - fail distributions that are far from uniform
 - what measure of distance?
 - L_1 distance: $|p-U|_1 = \sum |p_i - 1/n|$
 - L_2 distance: $|p-U|_2^2 = \sum (p_i - 1/n)^2$
- Interested in sample complexity in terms of n and bound on distance from uniform

Traditional methods

- Standard statistical techniques
 - χ^2 test
 - “Plug-in” estimates
- Require number of samples that is at least linear in n

Algorithm for L_1 distance

[Goldreich Ron] [Batu Fortnow R. Smith White]

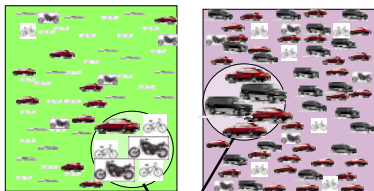
- Theorem: There is an algorithm which on input distribution p satisfies:
 - if $\|p-U\|_1 = 0$ output Pass
 - if $\|p-U\|_1 > \epsilon$ output Fail
 - sample, time complexity $\tilde{O}(n^{1/2} \epsilon^{-4})$
- Main idea: look at collisions!

Other examples of distribution properties

Testing closeness of two distributions:

Transactions of 20-30 yr olds

Transactions of 30-40 yr olds

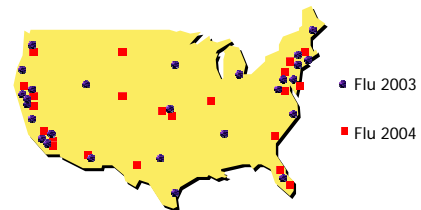


trend change?

Outbreak of diseases

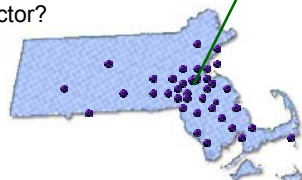


- Do they follow similar patterns?
- Are they correlated with income level, zip code?



Testing the monotonicity of distributions:

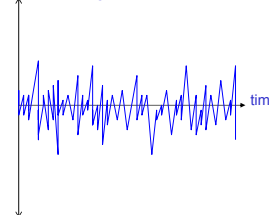
- Does the occurrence of cancer decrease with distance from the nuclear reactor?



Information in neural spike trails

[Strong, Koberle, de Ruyter van Steveninck, Bialek '98]

Neural signals



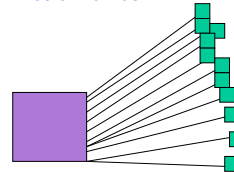
- Apply stimuli several times, each application gives sample of signal (spike trail)
- Study entropy of (discretized) signal to see which neurons respond to stimuli

Estimating Compressibility of Data

- General question undecidable
- Run-length encoding
- Huffman coding
 - Entropy
- Lempel-Ziv
 - “Color number” = Number of elements with probability at least $1/n$

Worm detection

- find “heavy hitters” – nodes that send to many distinct addresses
 - color number



What can't we do in sublinear time?

- local properties –
 - e.g., existence of element with factor of 10 change in probability from p to q
- fine estimation of distance to having property?
 - equivalently, *tolerant testing*

Some “easier” cases:

- Know something about the distribution
 - e.g., monotone, uniform over a subset
- Access to other kinds of queries

Properties of distributions:

- Are two distributions similar or very different?
 - One is uniform distribution [Goldreich Ron]
 - One is “known” distribution [Batu Fischer Fortnow Kumar Rubinfeld White]
 - General case [Batu Fortnow Rubinfeld Smith White]
- Approximate the entropy of a distribution [Batu Dasgupta Kumar Rubinfeld]
- Are two random variables independent? [Batu Fischer Fortnow Kumar Rubinfeld White]
- Is the distribution monotone? [Batu Kumar Rubinfeld]

Conclusions

- Sublinear time possible in many contexts
 - wide range of (open) problems
 - lots of techniques
 - simple algorithms, more involved analysis
 - what else can you compute in sublinear time?
- Other models:
 - use of a prover [Ergun Kumar Rubinfeld] [Batu Rubinfeld White][Dinur Reingold]
 - quantum property testing [Buhrman Fortnow Newman Roehrig]
- Related to work in sublinear/restricted space algorithms
 - streaming models
 - sketching (websites)