CPT-S 415

Big Data

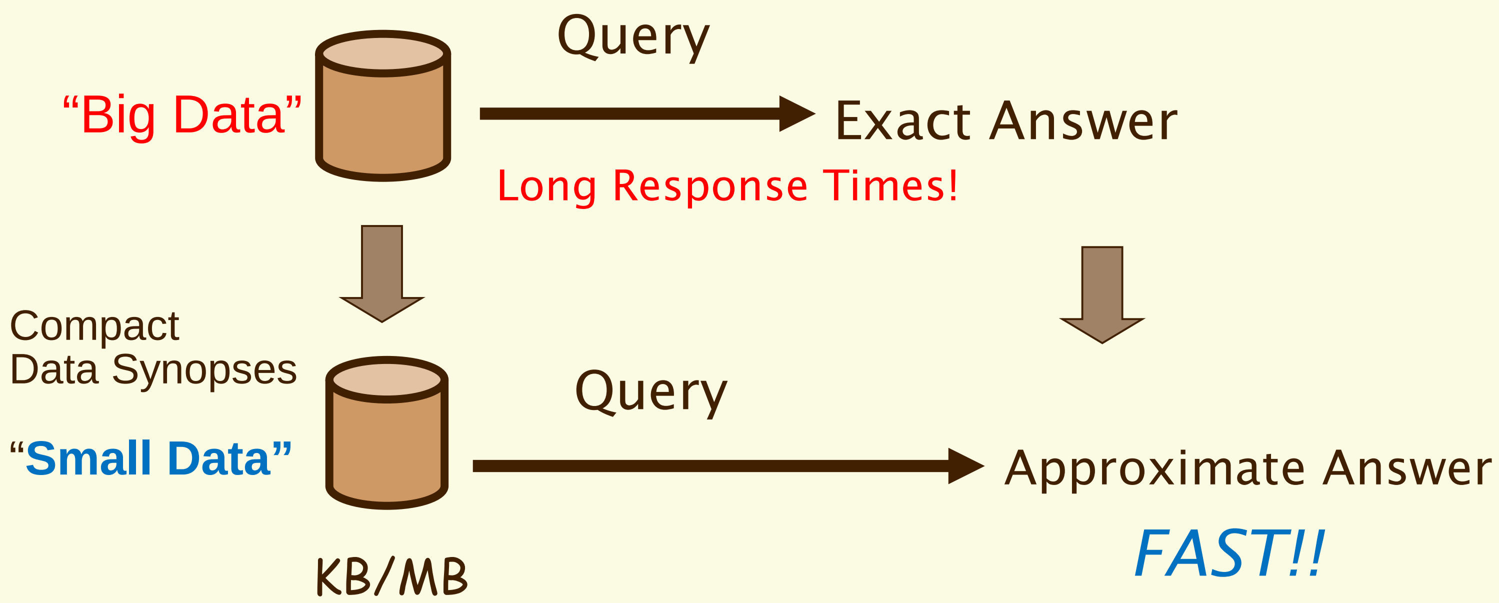
Yinghui Wu

EME B45

Approximate query processing

- ✓ Data-driven approximation
  - Data synopses: Histogram, Sampling, Wavelet
  - Graph synopses: Sketches, spanners, sparsifiers
- ✓ A principled search framework: Resource bounded querying

Data-driven Approximate Query Processing



How to construct effective *data synopses* ??

Histograms, samples, wavelets, sketches, spanners, sparsifiers

Histograms

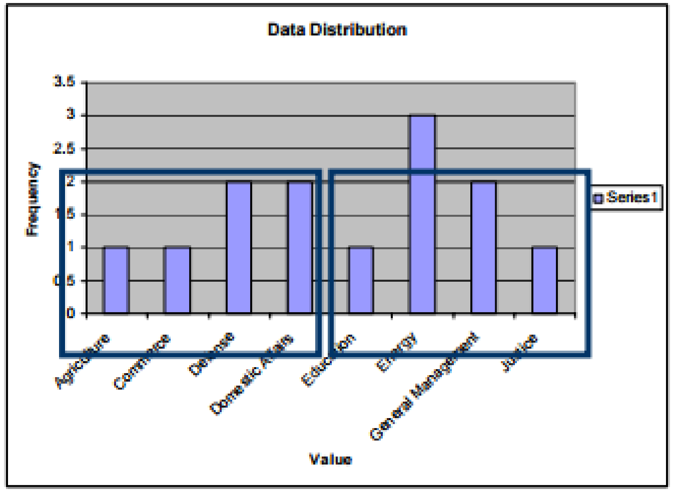
- **Partition attribute value(s) domain into a set of buckets**
- Estimation of data distribution (mostly for aggregation); approximate the frequencies in each bucket in common fashion
- Equi-width, equi-depth, V-optimal

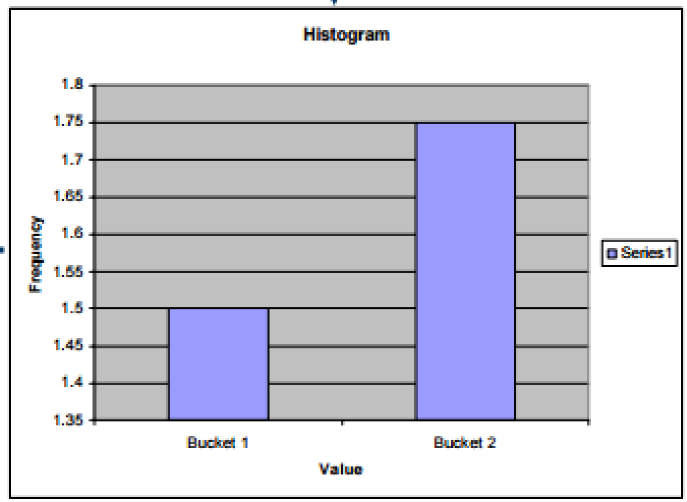
Name	Salary	Department
Zeus	100K	General Management
Poseidon	80K	Defense
Pluto	80K	Justice
Aris	50K	Defense
Ermis	60K	Commerce
Apollo	60K	Energy
Hefestus	50K	Energy
Hera	90K	General Management
Athena	70K	Education
Aphrodite	60K	Domestic Affairs
Demeter	60K	Agriculture
Hestia	50K	Domestic Affairs
Artemis	60K	Energy

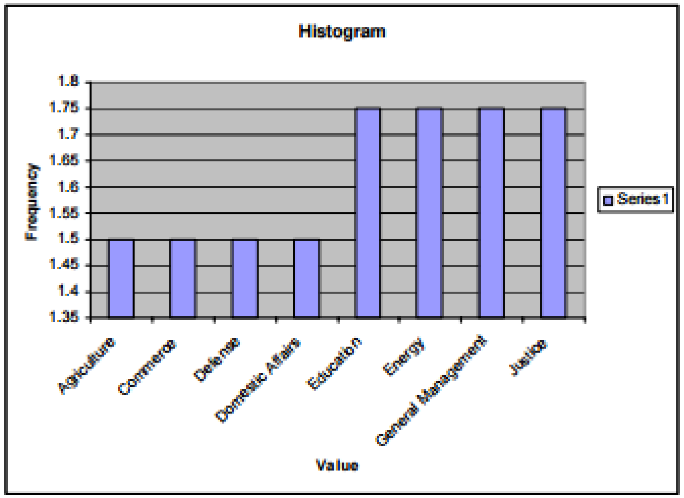
Department	Frequency
General Management	2
Defense	2
Education	1
Domestic Affairs	2
Agriculture	1
Commerce	1
Justice	1
Energy	3

From data distribution to histogram

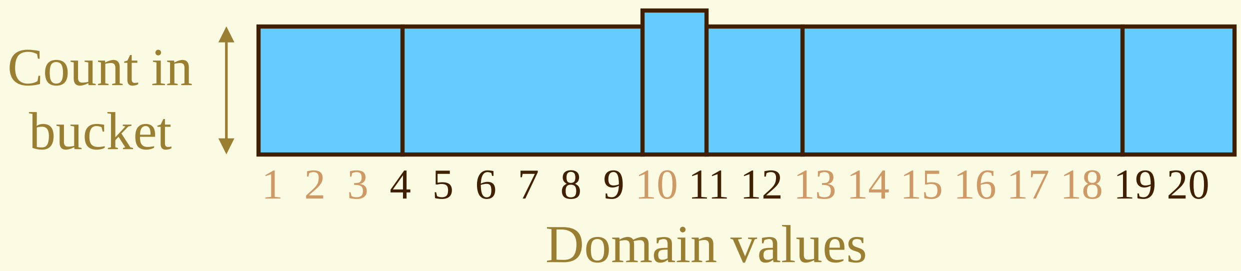
Department	Histogram H1	
	Frequency in Bucket	Approximate Frequency
Agriculture	1	1.5
Commerce	1	1.5
Defense	2	1.5
Domestic Affairs	2	1.5
Education	1	1.75
Energy	3	1.75
General Management	2	1.75
Justice	1	1.75







Equi-Depth



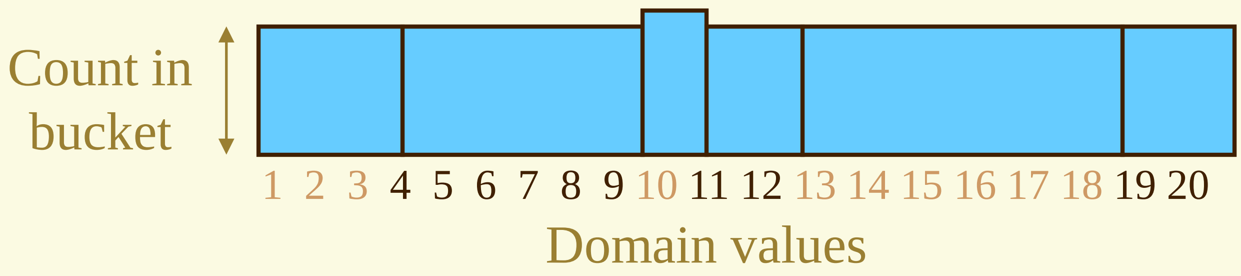
- ✓ Goal: Equal number of rows per bucket (B buckets in all)
- ✓ Can **construct** by first sorting then taking B-1 equally-spaced splits

1 2 2 3 4 7 8 9 10 10 10 10 11 11 12 12 14 16 16 18 19 20 20 20

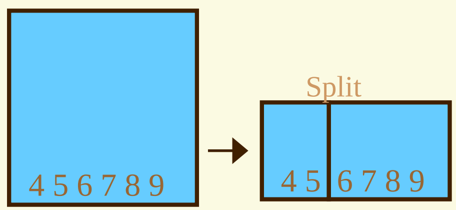
- ✓ **Faster construction:** Sample & take equally-spaced splits in sample
  - Nearly equal buckets
  - faster algorithm: one-pass quantile

    "Space-Efficient Online Computation of Quantile Summaries", Michael Greenwald, et al., SIGMOD 01

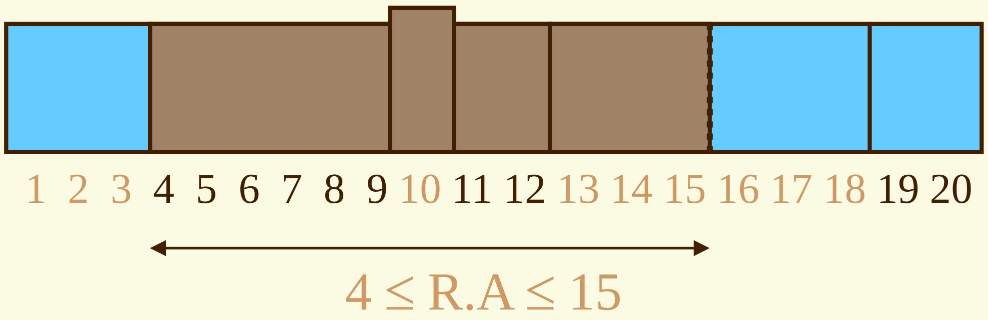
Equi-Depth



- ✓ Can **maintain** using one-pass algorithms (insertions only), or
- ✓ Maintain a larger sample on disk in support of histogram maintenance
  - Keep histogram bucket counts up-to-date by incrementing on row insertion, decrementing on row deletion
  - Merge adjacent buckets with small counts
  - Split any bucket with a large count, using the sample to select a split value, i.e, take median of the sample points in bucket range



Answering Queries: Equi-Depth



Answering queries:

- select count(*) from R where 4 <= R.A <= 15
- approximate answer: F * |R|/B, where
  - F = number of buckets, including fractions, that overlap the range
- Answer: 3.5* 24/6 = 14; actual count: 13
  - error? 0.5*24/6 = 2

Answering queries from 1-D histograms (in general):
(Implicitly) map the histogram back to an approximate relation,
& apply the query to the approximate relation

Sampling: Basics

$$AVERAGE_{estimated} = AVERAGE_{sample\_set}$$

$$COUNT_{estimated} = COUNT_{sample\_set} \times \frac{N}{n}$$

$$SUM_{estimated} = SUM_{sample\_set} \times \frac{N}{n} = AVERAGE_{sample\_set} \times N$$

$$MAX_{estimated} = MAX_{sample\_set} \qquad MIN_{estimated} = MIN_{sample\_set}$$

Normal query:

```
SELECT count(*),sum(sales),
average(sales) , ...
FROM  SalesFact, Time, ...
WHERE joins and restrictions
GROUP BY group conditions
```

Rewritten query:

```
SELECT sum(1/SP),sum(sales/SP),
sum(sales/SP)/sum(1/SP), ...
FROM SW_SalesFact, SW_Time, ...
WHERE joins and restrictions
GROUP BY group conditions
```

- Leverage extensive literature on confidence intervals for sampling
  - Actual answer is within the interval [a,b] with a given probability
    - E.g., 54,000 ± 600 with prob ≥ 90%

Sampling: Confidence Intervals

Method	90% Confidence Interval (±)	Guarantees?
Central Limit Theorem	$1.65 * \sigma(S) / \mathrm{sqrt}(\lvert S \rvert)$	as $\lvert S \rvert \to \infty$
Hoeffding	$1.22 * (\mathrm{MAX}-\mathrm{MIN}) / \mathrm{sqrt}(\lvert S \rvert)$	always
Chebychev (known $\sigma(R)$)	$3.16 * \sigma(R) / \mathrm{sqrt}(\lvert S \rvert)$	always
Chebychev (est. $\sigma(R)$)	$3.16 * \sigma(S) / \mathrm{sqrt}(\lvert S \rvert)$	as $\sigma(S) \to \sigma(R)$

Confidence intervals for Average: select avg(R.A) from R

$\sigma(R)$ = standard deviation of the values of R.A; $\sigma(S)$ = s.d. for S.A

- ✓ If predicates, S above is subset of sample that satisfies the predicate
- ✓ Quality of the estimate depends only on the variance in R & |S| after the predicate: So 10K sample may suffice for 10B row relation!
  - Advantage of larger samples: can handle more selective predicates

One-Pass Uniform Sampling

- Best choice for incremental maintenance
  - Low overheads, no random data access
- Reservoir Sampling [Vit85]: Maintains a sample S of a fixed-size k
  http://www.cs.umd.edu/~samir/498/vitter.pdf
  - Add each new item to S with probability M/N, where N is the current number of data items
  - If add an item, evict a random item from S
  - Instead of flipping a coin for each item, determine the number of items to skip before the next to be added to S

```
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i = 1 to k
      R[i] := S[i]

  // replace elements with gradually decreasing probability
  for i = k+1 to n
    j := random(1, i)   // important: inclusive range
    if j <= k
        R[j] := S[i]
```

Wavelets

- In signal processing community, wavelets are used to break the complicated signal into single component.
- Similarly in approximate query processing, wavelets are used to break the dataset into simple component.
- Haar wavelet - simple wavelet, easy to understand

One-Dimensional Haar Wavelets

- ✓ Wavelets: mathematical tool for hierarchical decomposition of functions/signals
- ✓ Haar wavelets: simplest wavelet basis, easy to understand and implement
  - *Recursive pairwise averaging and differencing* at different resolutions

Resolution	Averages	Detail Coefficients
3	[2, 2, 0, 2, 3, 5, 4, 4]	----
2	[2, 1, 4, 4]	[0, -1, -1, 0]
1	[1.5, 4]	[0.5, 0]
0	[2.75]	[-1.25]

[2.75, -1.25, 0.5, 0, 0, -1, -1, 0]

Haar wavelet decomposition

Haar Wavelet Coefficients

- Using wavelet coefficients one can pull the raw data
- Keep only the large wavelet coefficients and pretend other coefficients to be 0.

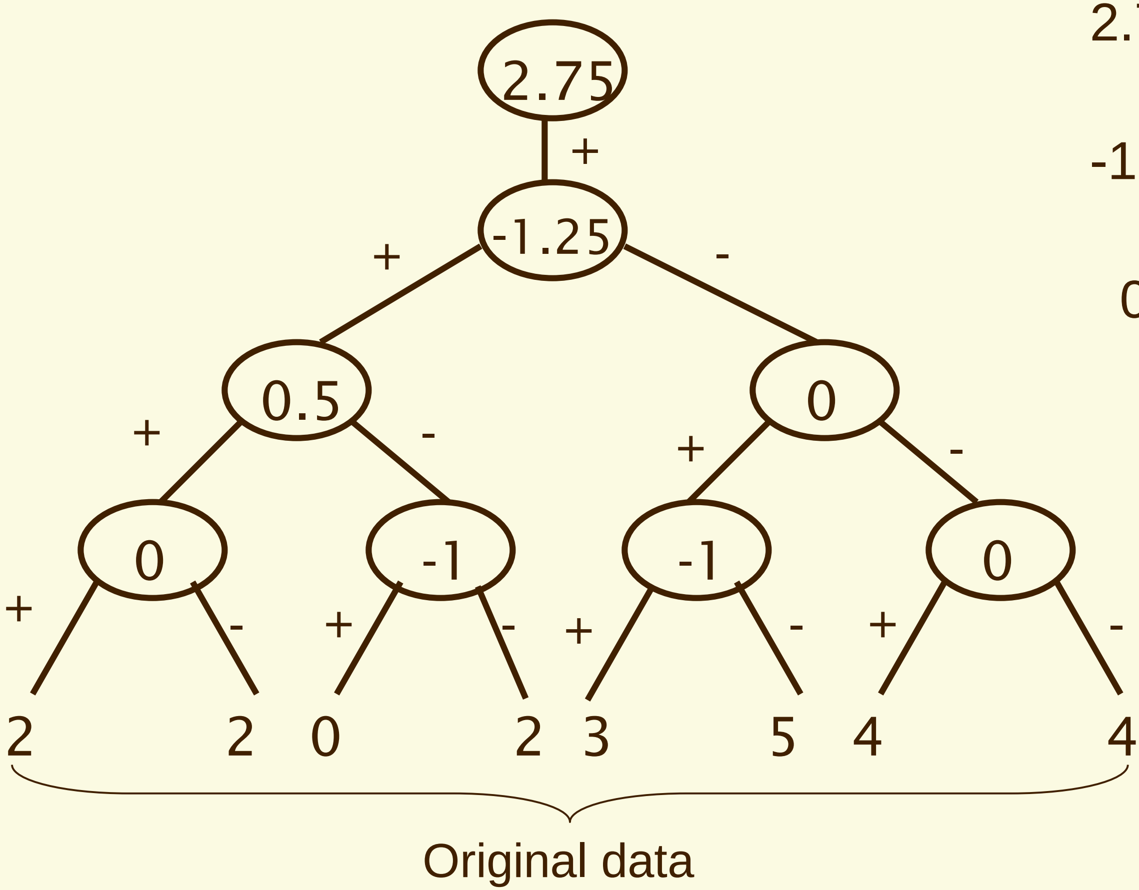
[2.75, -1.25, 0.5, 0, 0, -1, -1, 0]

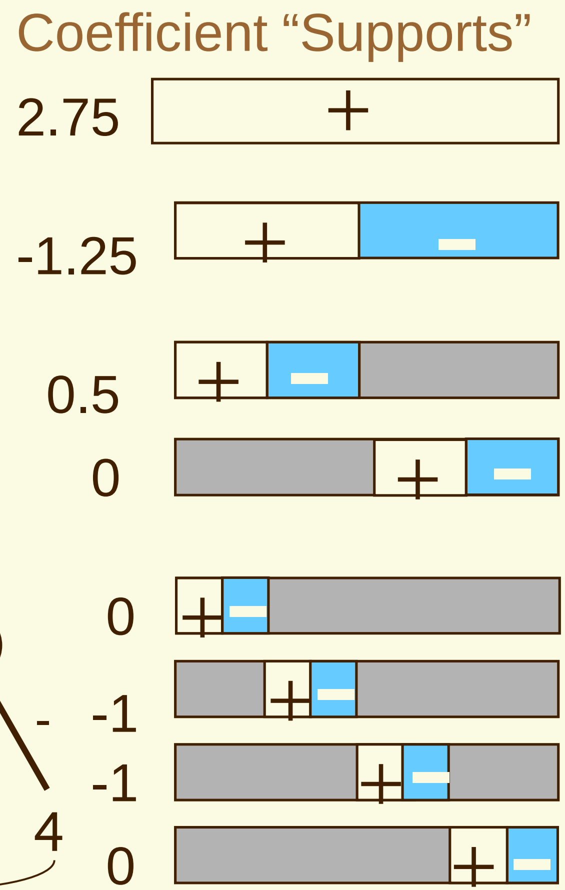
[2.75, -1.25, 0.5, 0, 0, 0, 0, 0]-synopsis of the data

- The elimination of small coefficients introduces only small error when reconstructing the original data

Haar Wavelet Coefficients

- Hierarchical decomposition structure (a.k.a. "error tree")





Example

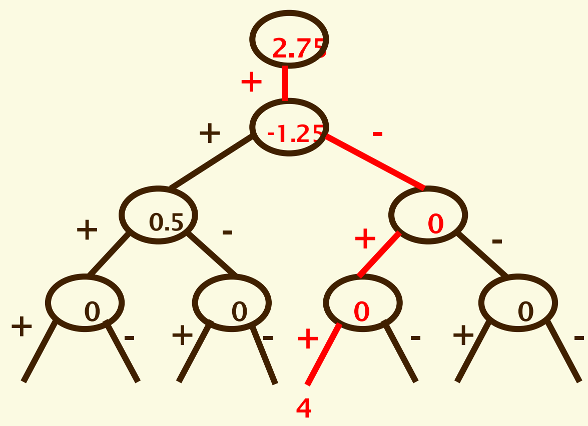
Query: SELECTsalary
FROM employee
WHERE empid=5

Result: By using the synopsis [2.75,- 1.25, 0.5, 0, 0, 0, 0, 0] and constructing the tree on fly, salary=4 will be returned, whereas the correct result is salary=3.

This error is due to truncation of wavelength

Employee

Empid	Salary
1	2
2	2
3	0
4	2
5	3
6	5
7	4
8	4



Example-2 on range query

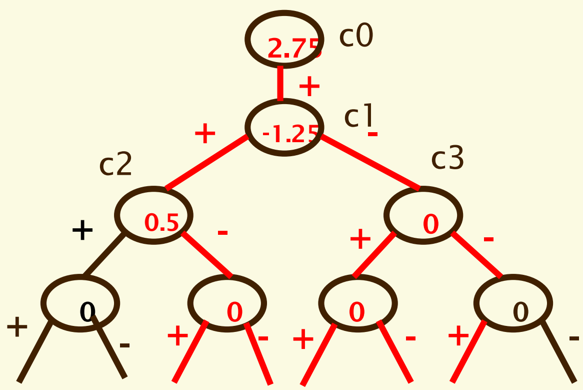
- SELECT sum(salary) FROM Employee WHERE 3 <= empid <=7
- Find the Haar wavelet transformation and construct the tree
- $A(l:h) = \sum_{c_j \in \mathtt{path}(A[l]) \cup \mathtt{path}(A[h])} x_j,$ where

$$x_j = \begin{cases} (h - l + 1) \cdot c_j, \text{ if } j = 0 \\ (|\mathtt{leftleaves}(c_j, l:h)| - |\mathtt{rightleaves}(c_j, l:h)|) \cdot c_j, \text{ otherwise.} \end{cases}$$

- Result: $A(2:6) = 5c_0 + (2-3)c_1 - 2c_2 =$ (6-2+1)*2.75 + (2-3)* (-1.25) – 2*0.5 = 14

Employee

Empid	Salary
1	2
2	2
3	0
4	2
5	3
6	5
7	4
8	4



Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches, Graham, et.al http://db.ucsd.edu/static/Synopses.pdf

Comparison with sampling

- ✓ For Haar wavelet transformation all the data must be numeric. In the example, even empid must be numeric and must be sorted
- ✓ Sampling gives the probabilistic error measure whereas Haar wavelet does not provide any
- ✓ Haar wavelet is more robust than sampling. The final average gives the average of all data values. Hence all the tuples are involved.

Graph data synopses

Graph Data Synopses

- ✓ Graph synopses
  - Small synopses of big graphs
  - Easy to construct
  - Yield good approximations of the relevant properties of the data set
- ✓ Types of synopses
  - Neighborhood sketches
  - Graph Sampling
  - Sparsifiers
  - Spanners
  - Landmark vectors
  - …

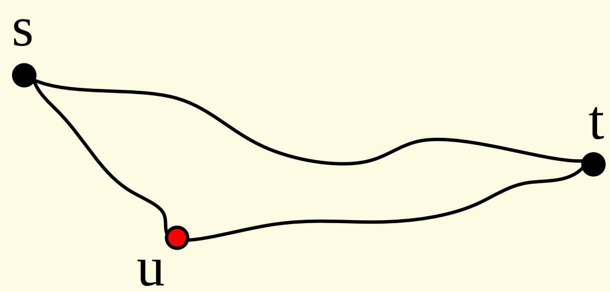
Landmarks for distance queries

- Offline
  - Precompute distance of all nodes to a small set of nodes (landmarks)
  - Each node is associated with a vector with its SP-distance from each landmark (embedding)
- Query-time
  - $d(s,t) = ?$
  - Combine the embeddings of s and t to get an estimate of the query

Algorithmic Framework

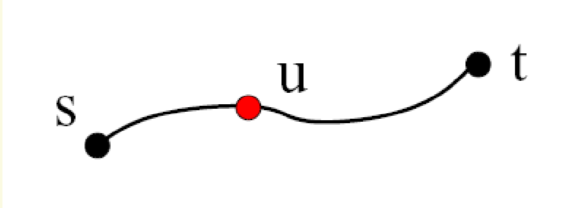
- Triangle Inequality

$$d_G(s,t) \le d_G(s,u) + d_G(u,t),$$
$$d_G(s,t) \ge |d_G(s,u) - d_G(u,t)|$$

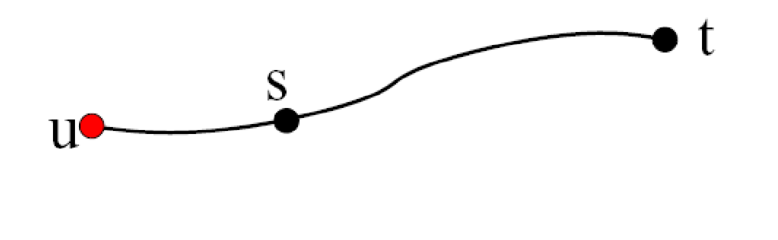


- Observation: the case of equality

$$d_G(s,t) \ = \ d_G(s,u) + d_G(u,t)$$



$$d_G(s,t) \ = \ |d_G(s,u) - d_G(u,t)|$$



Example query: d(*s*,*t*)

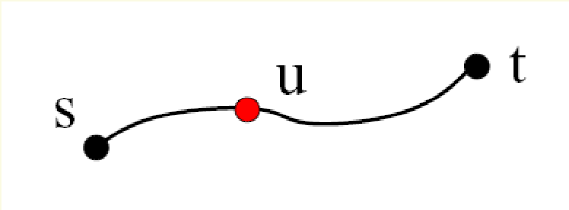
	d(_,u_1)	*d*(_,u_2)	d(_,u_3)	d(_,u_4)
s	2	4	5	2
t	3	5	1	4

UB	**5**	9	6	6
LB	1	1	**4**	2

$$\max_i |s_i - t_i| \leq d_G(s,t) \leq \min_j \{s_j + t_j\}$$

Coverage Using Upper Bounds

- A landmark u covers a pair (s, t), if u lies on a shortest path from s to t
- Problem Definition : find a set of k landmarks that cover as many pairs (s,t) in $V \times V$
  - NP-hard
  - $k = 1$: node with the highest betweenness centrality
  - $k > 1$: greedy set-cover (too expensive)



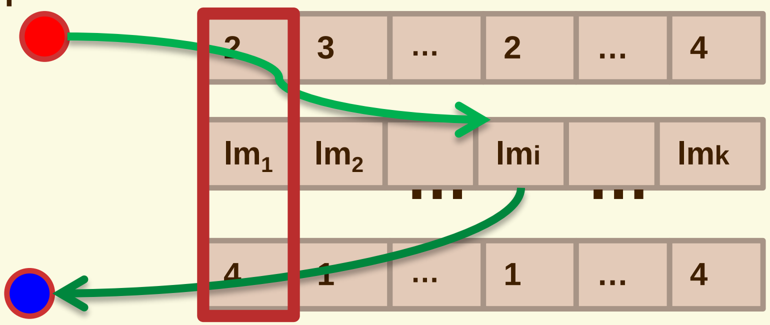
- How to select? Random; high centrality; high degree, high Pagerank scores…

The Landmark Method

1. Selection: Select *k* landmarks

2. Offline: Run *k* BFS/Dijkstra and store the embeddings of each node:

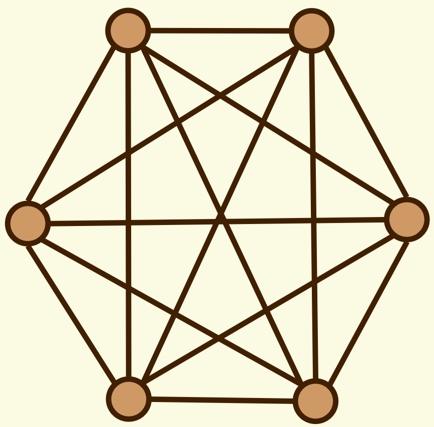
   $\Phi(s) = \langle d_G(u_1, s), d_G(u_2, s), \ldots, d_G(u_k, s)\rangle$
   $= \langle s_1, s_2, \ldots, s_d\rangle$

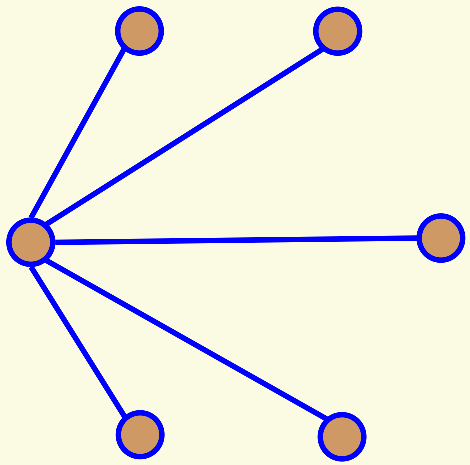


3. Query-time: $d_G(s,t) = ?$
   – Fetch $\Phi(s)$ and $\Phi(t)$
   – Compute $\min_i\{s_i + t_i\}$... in time $O(k)$

Spanners

- Let G be a weighted undirected graph.
- A subgraph H of G is a t-spanner of G iff $\forall u,v \in G$, $\delta_H(u,v) \leq t\ \delta_G(u,v)$.
- The smallest value of t for which H is a t-spanner for G is called the **stretch factor** of H .





How to construct a spanner?

<u>Input</u> : A weighted graph G,

A positive parameter r.

The weights need not be unique.

<u>Output</u> : A sub graph G'.

Step 1: Sort E by non-decreasing weight.

Step 2: Set G' = { }.

Step 3: For every edge e = [u,v] in E, compute P(u,v), the shortest path from u to v in the current G'.

Step 4: If, r.Weight(e) < Weight(P(u,v)),

add e to G',

else, reject e.

Step 5:Repeat the algorithm for the next edge in E and so on.

Approximate Distance Oracles

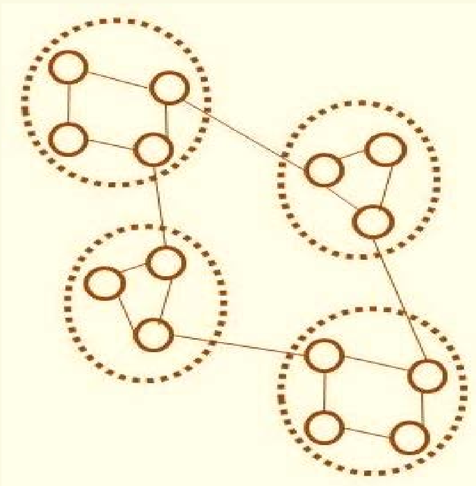
- Consider a graph **G=(V,E)**. An approximate distance oracle with a stretch **k** for the graph **G** is a data-structure that can answer an approximate distance query for any two vertices with a stretch of at most **k**.
- For every **u,v** in **V** the data structure returns in "short time" an approximate distance **d'** such that:

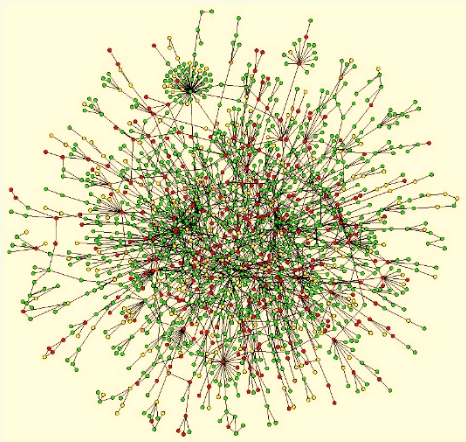
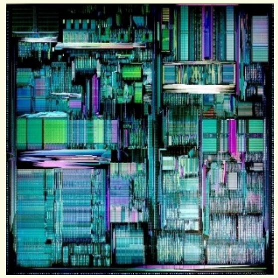
$$d_G(u,v) \le d' \le k \cdot d_G(u,v)$$

- A k-spanner is a k distance oracle
- Theorem: One can efficiently find a (2k-1)-spanner with at most $n^{1+1/k}$ edges.

Graph Sparsification

- ✓ Is there a simple pre-processing of the graph to reduce the edge set that can "clarify" or "simplify" its cluster structure?
- ✓ Application: graph community detection



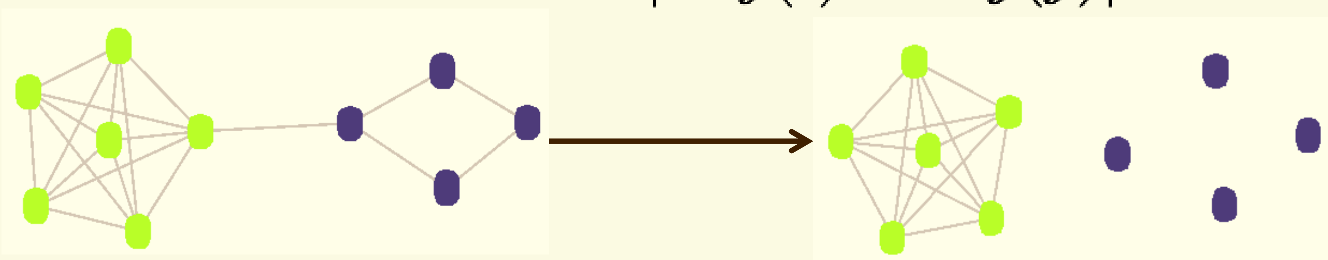



Global Sparsification

✓ Parameter: Sparsification ratio, *s*

- For each edge <i,j>: calculate Sim (<i,j>)
- Retain top s% of edges in order of Sim, discard others

$$Sim(< i, j >) = \frac{|Adj(i) \cap Adj(j)|}{|Adj(i) \cup Adj(j)|}$$



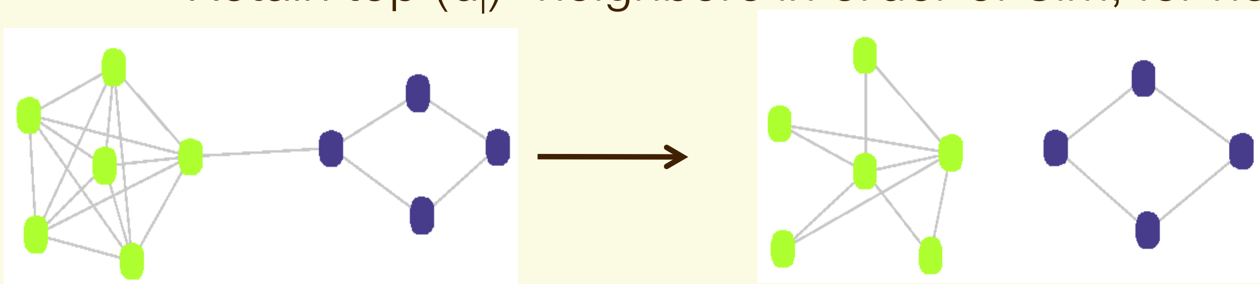
Dense clusters: over-represented;
sparse clusters: under-represented

Local Sparsification

- ✓ "Local Graph Sparsification for Scalable Clustering", Venu Satulur et.al, SIGMOD 11

- ✓ Parameter: Sparsification exponent, e ($0 < e < 1$)

- ✓ For each node i of degree d_i:
  - For each neighbor j:
    Calculate Sim (<i,j>)
  - Retain top $(d_i)^e$ neighbors in order of Sim, for node i



Ensures representation of clusters of varying densities

Applications of Sparsifiers

- Faster (1 ± ε)-approximation algorithms for flow-cut problems
  - Maximum flow and minimum cut [Benczur-Karger 02, …, Madry 10]
  - Graph partitioning [Khandekar-Rao-Vazirani 09]
- Improved algorithms for linear system solvers [Spielman-Teng 04, 08]
- Sample each edge with a certain probability
  - Non-uniform probability chosen captures "importance" of the cut (several measures have been proposed)
  - Distribution of the number of cuts of a particular size [Karger 99]
  - Chernoff bounds

Data-driven approximation for bounded resources

The approximation theory revisited

- ✓ Traditional approximation algorithms $\mathcal{T}$: for an NPO
  - for each instance x, $\mathcal{T}$(x) computes a feasible solution y
  - quality metric f(x, y)
  - performance ratio (minimization): for all x,

$$OPT(x) \leq f(x, y) \leq \eta \, OPT(x)$$

Big data?

- ✓ Approximation: for even low PTIME problems, not just NPO
- ✓ Quality metric: answer to a query is not necessarily a number
- ✓ Approach: it does not help much if $\mathcal{T}$(x) conducts computation on "big" data x directly!

A quest for revising approximation algorithms for querying big data

Data-driven: Resource bounded query answering

✓ Input: A class Q of queries, a resource ratio $\alpha \in [0, 1)$, and a performance ratio $\eta \in (0, 1]$

✓ Question: Develop an algorithm that given any query $Q \in$ Q and dataset D,

- *accesses a fraction* **D_α** *of D such that* $|D_\alpha| \le \alpha|D|$
- computes as Q(**D_α**) as approximate answers to Q(D), and
- *accuracy(Q, D,* α) $\geq \eta$

$$
\begin{tikzcd}
\mathbf{D} \arrow[rr, "\text{dynamic reduction}"] \arrow[d, "Q"'] & & \mathbf{D}_\alpha \arrow[d, "Q'"] \\
Q(\,\mathbf{D}\,) & & Q'(\,\mathbf{D}_\alpha\,) \arrow[ll, "\text{approximation}"']
\end{tikzcd}
$$

Accessing $\alpha|D|$ amount of data in the entire process

Resource bounded query answering

$$\begin{tikzcd}[column sep=huge, row sep=large]
\mathbf{D} \arrow[r, "\text{dynamic reduction}"] \arrow[d, "Q"'] & \mathbf{D}_\alpha \arrow[d, "Q"] \\
Q(\mathbf{D}) & Q(\mathbf{D}_\alpha) \arrow[l, "\text{approximation}"']
\end{tikzcd}$$

- ✓ Resource bounded: resource ratio $\alpha \in [0, 1)$
  - decided by our available resources: time, space, energy…
- ✓ Dynamic reduction: given Q and D
  - – *find* D_α *for all Q*
  - histogram, wavelets, sketches, sampling, …

In combination with other tricks for making big data small

Personalized social search

Graph Search, Facebook

- ✓ Find me all my friends who live in Seattle and like cycling
- ✓ Find me restaurants in London my friends have been to
- ✓ Find me photos of my friends in New York

Localized patterns

personalized social search with α = 0.0015%!

with near 100% accuracy

- ✓ *$1.5 * 10^{-6} * 1PB\ (10^{15}B/10^{6}GB) = 15 * 10^{9} = 15GB$*
- ✓ *We are making big graphs of PB size as small as 15GB*

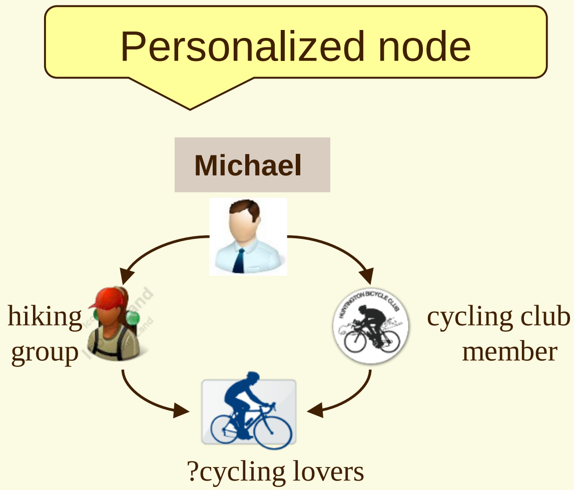
Add to this data synopses, schema, distributed, views, …

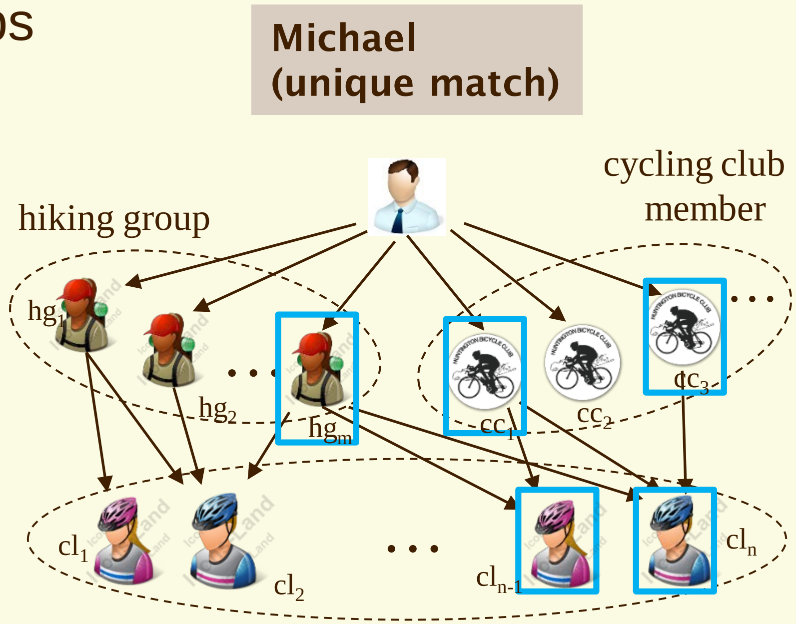
make big graphs of PB size fit into our memory!

Localized queries

Localized queries: can be answered locally
- Graph pattern queries: revised simulation queries
  - matching relation over d_Q-neighborhood of a personalized node

Michael: "find cycling fans who know both my friends in cycling club and my friends in hiking groups

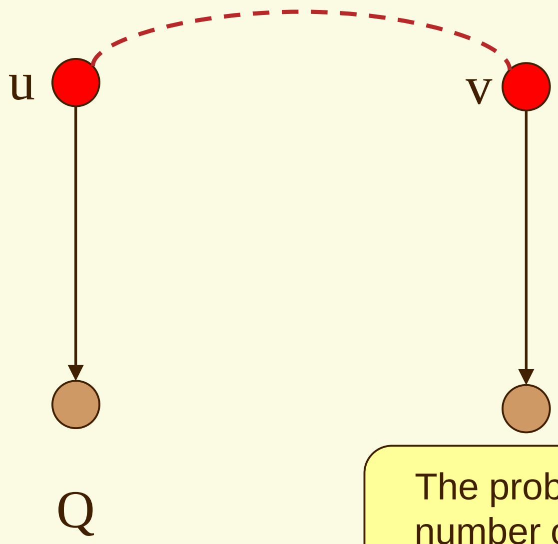




Personalized social search, ego network analysis, …

Resource-bounded simulation

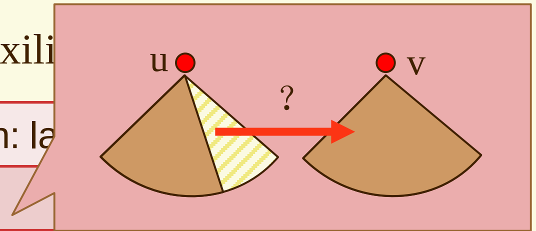
Preprocessing (auxiliary information) → dynamic reduction (compute reduced subgraph) → Approximate query evaluation over reduced subgraph



local auxiliary info

degree	|neighbor|	

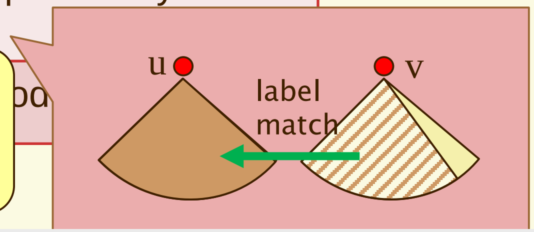
If v is included, the number of additional nodes that need also to be included – budget



Dynamically updated auxili

Boolean guarded condition: la
Cost c(u,v)
Potential p(u,v), estimated probability that v matches u

The probability for v to match u (total number of nodes in the neighbor of v that are candidate matches



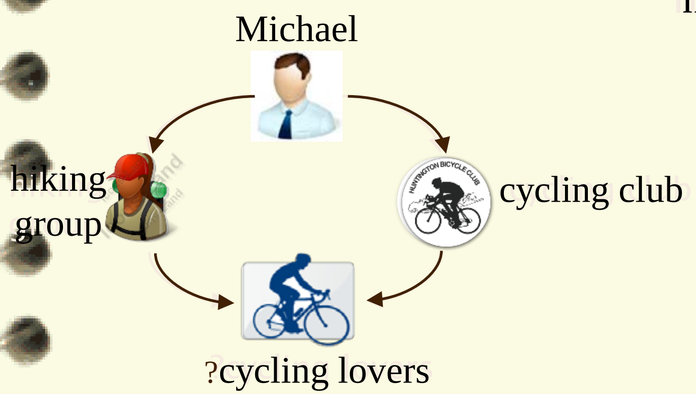
Query guided search – potential/cost estimation

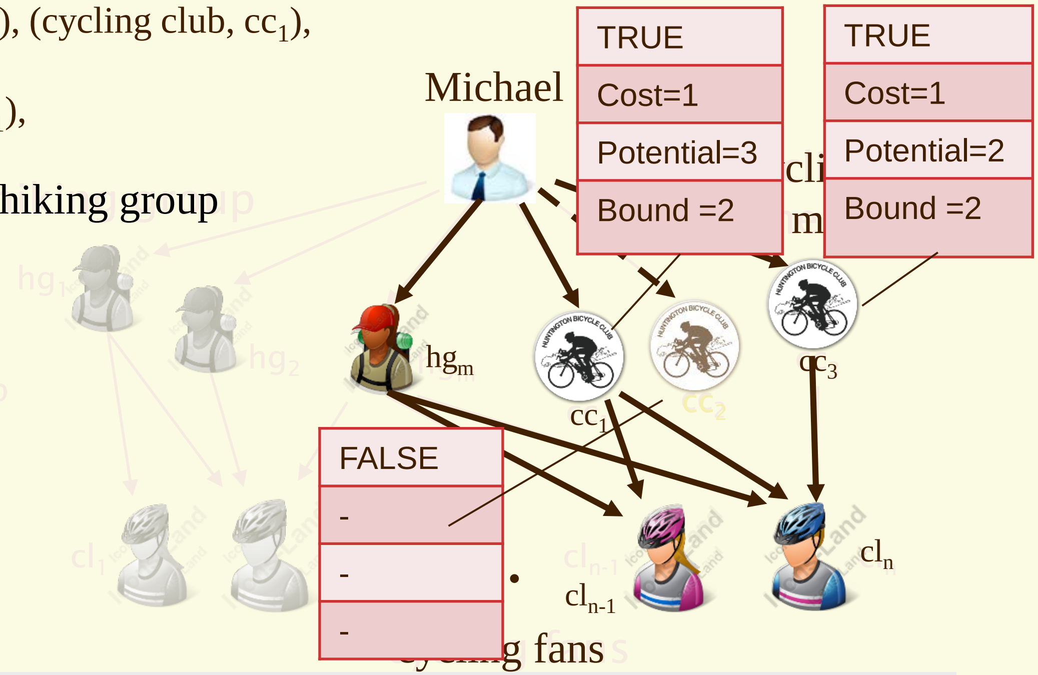
Resource-bounded simulation

preprocessing → dynamic reduction (compute reduced subgraph) → Approximate query evaluation over reduced subgraph

bound = 14
visited = 16

Match relation:
(Michael, Michael),
(hiking group, hg_m), (cycling club, cc_1),
(cycling club, cc_3),
(cycling lover, cl_{n-1}),
(cycling lover, cl_n)





Dynamic data reduction and query-guided search

Summing up

Approximate query answering

Challenges: to get real-time answers

✓Big data and costly queries

✓Limited resources

Two approaches:

✓Query-driven approximation

- Cheaper queries
- Retain sensible answers

Combined with techniques for making big data small

✓Data-driven approximation

Reduce data of PG size to GB

- 6 type of data synopses construction methods
  (histogram, sample, wavelet, sketch, spanner, sparsifier)
- Dynamic data reduction
- Query-guided search

Query big data within bounded resources

Summary and review

- What is query-driven approximation? When can we use the approach?
- Traditional approximation scheme does not work very well for query answering in big data. Why?
- What is data-driven dynamic approximation? Does it work on localized queries? Non-localized queries?
- What is query-guided search?

- *Think about the algorithm you will be designing for querying large datasets. How can approximate querying idea applied?*

Papers for you to review

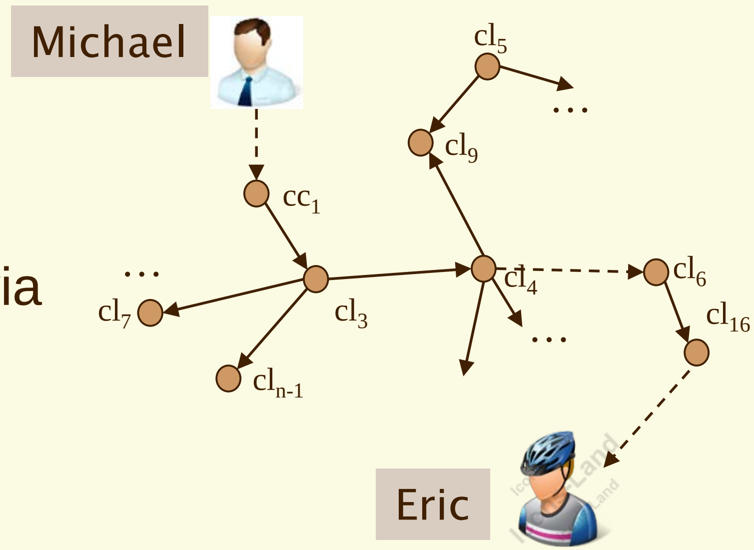
- G. Gou and R. Chirkova. Efficient algorithms for exact ranked twig-pattern matching over graphs. In SIGMOD, http://dl.acm.org/citation.cfm?id=1376676
- H. Shang, Y. Zhang, X. Lin, and J. X. Yu. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. PVLDB,2008. http://www.vldb.org/pvldb/1/1453899.pdf
- R. T. Stern, R. Puzis, and A. Felner. Potential search: A bounded-cost search algorithm. In ICAPS, 2011. (search Google Scholar)
- S. Zilberstein, F. Charpillet, P. Chassaing, et al. Real-time problem solving with contract algorithms. In IJCAI, 1999. (search Google Scholar)
- W. Fan, X. Wang, and Y. Wu. Diversified Top-k Graph Pattern Matching, VLDB 2014. (query-driven approximation)
- W. Fan, X. Wang, and Y. Wu. Querying big graphs with bounded resources, SIGMOD 2014. (data-driven approximation)

Non-localized queries

- Reachability
  - Input: A directed graph G, and a pair of nodes s and t in G
  - Question: Does there exist a path from s to t in G?

Non-localized: t may be far from s

Is Michael connected to Eric via social links?



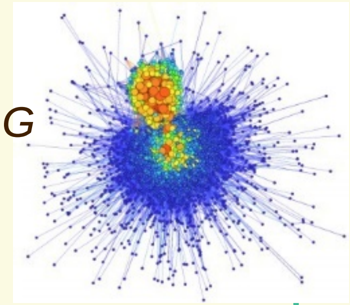
Does dynamic reduction work for non-localized queries?

Resource-bounded reachability

<u>accuracy measure</u>

$$precs = \frac{tp}{tp + fp} \quad recall = \frac{tp}{tp + fn}$$

big graph G



Reachability query

results

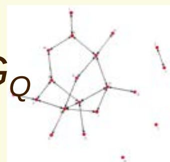
O(|G|)

Approximation (experimentally Verified; no false positive, in time **$O(\alpha|G|)$**)

Reduction

size $|G_Q| \le \alpha|G|$

small tree index G_Q



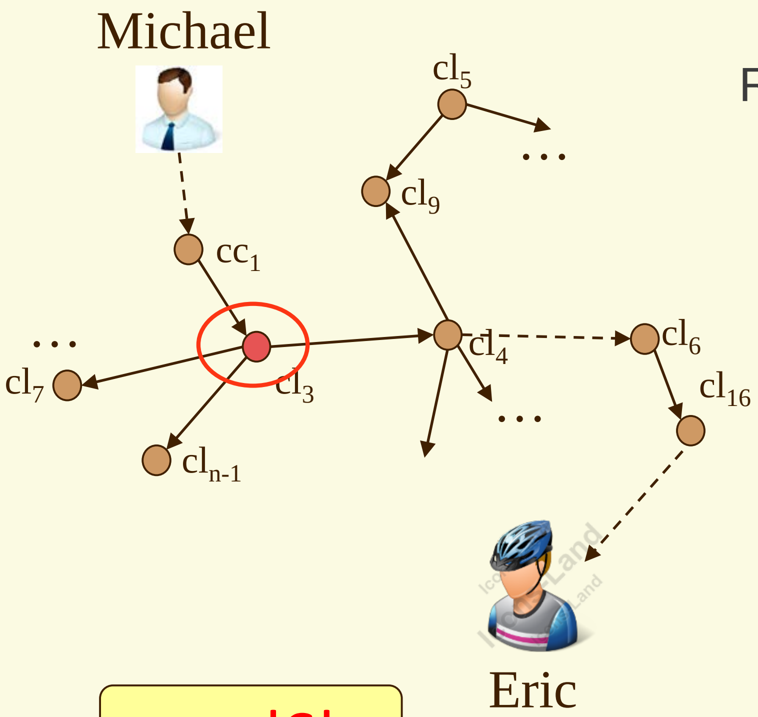
Reachability query

results

dynamic reduction for non-localized queries

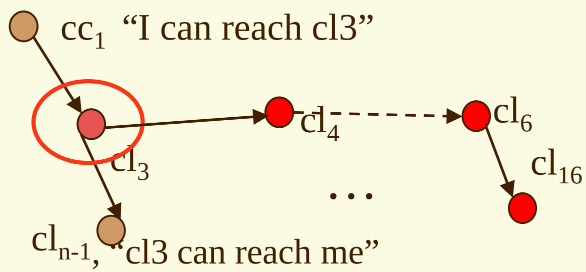
Preprocessing: landmarks

Preprocessing → dynamic reduction (compute landmark index) → Approximate query evaluation over landmark index



Recall Landmarks

- a landmark node covers certain number of node pairs
- Reachability of the pairs it covers can be computed by landmark labels



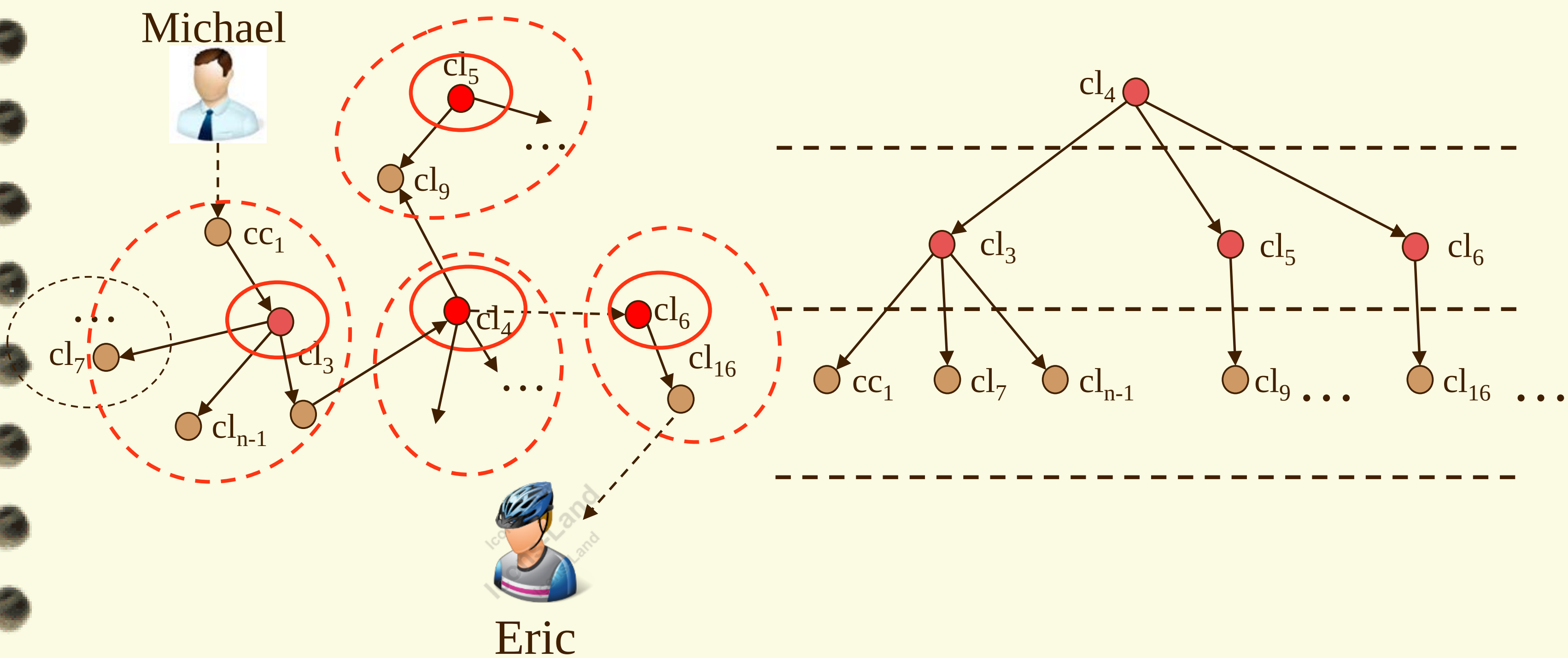
<= α|G|

Search landmark index instead of G

Hierarchical landmark Index

Landmark Index

- landmark nodes are selected to encode pairwise reachability
- Hierarchical indexing: apply multiple rounds of landmark selection to construct a tree of landmarks

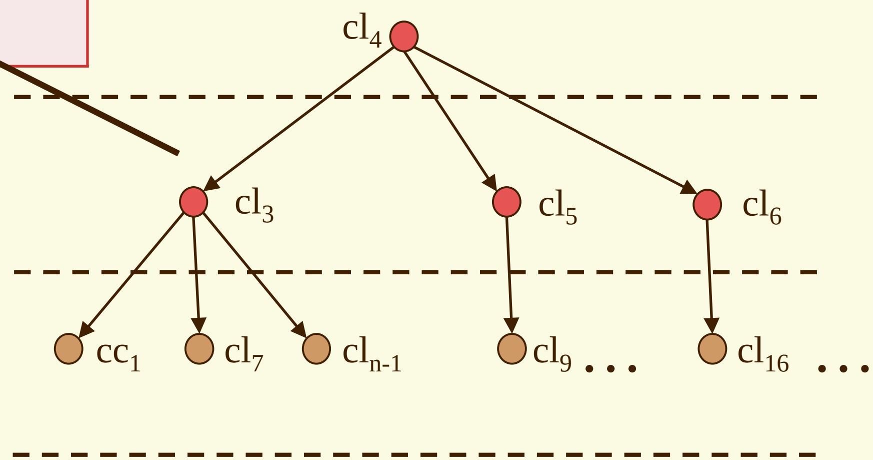


v can reach v' if there exists v1, v2, v2 in the index such that v reaches v1, v2 reaches v', and v1 and v2 are connected to v3 at the same level

Hierarchical landmark Index

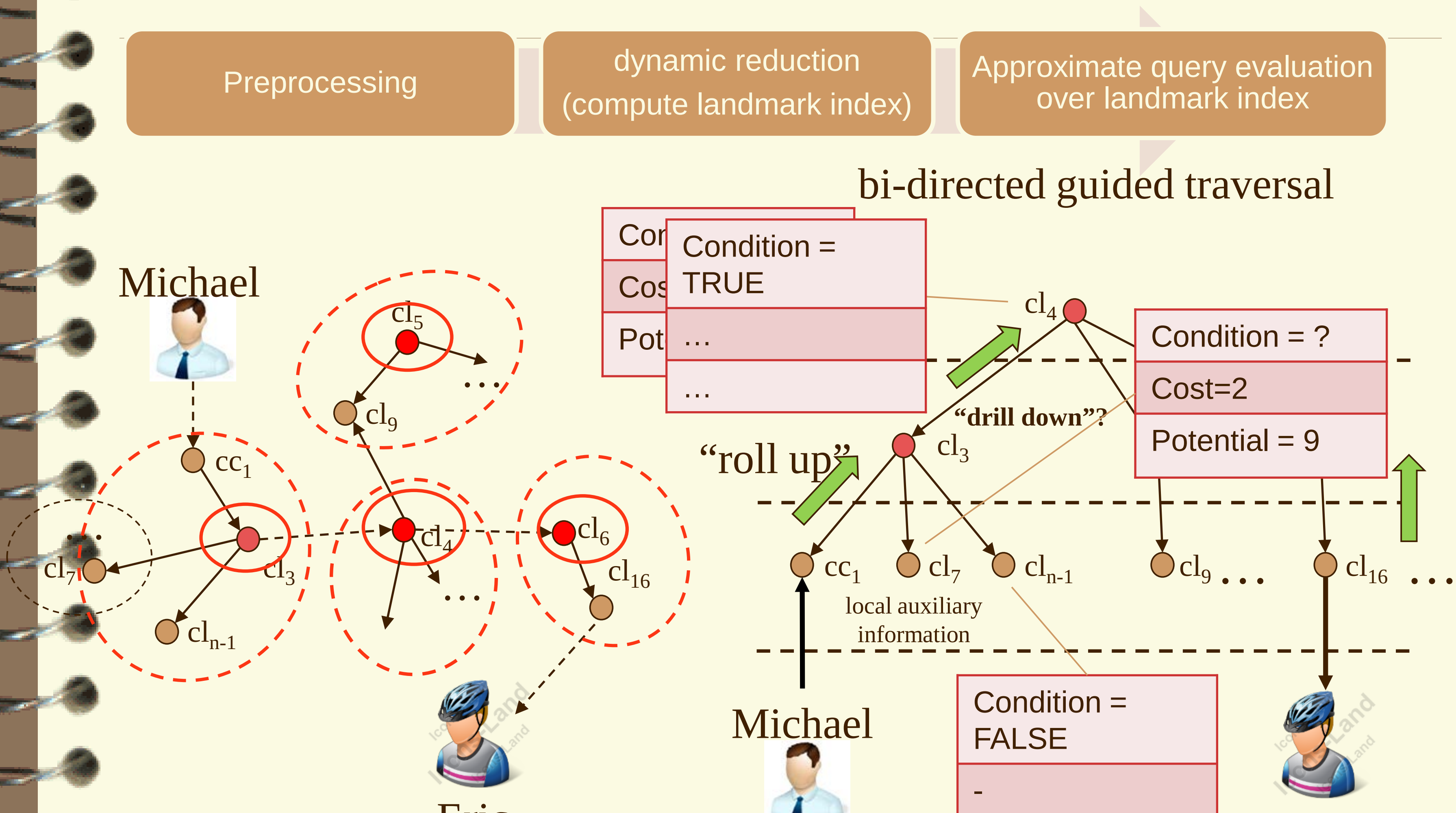
Boolean guarded condition (v, vp, v')
Cost c(v): size of unvisited landmarks in the subtree rooted at v
Potential P(v), total cover size of unvisited landmarks as the children of v

Cover size
Landmark labels/encoding
Topological rank/range



Guided search on landmark index

Resource-bounded reachability



Drill down and roll up