Rethink Possible

Structure-Aware Sampling:

Flexible and Accurate Summarization

Edith Cohen, Graham Cormode, Nick Duffield

AT&T Labs-Research

Summaries and Sampling

- Approximate summaries are vital in managing large data
  - E.g. sales records of a retailer; network activity for an ISP
  - Need to store compact summaries for later analysis
- State-of-the-art summarization via sampling
  - Widely deployed in many settings
  - Models data as (key, weight) pairs
  - General purpose summary, enables subset-sum queries
  - Higher level analysis: quantiles, heavy hitters, other patterns & trends

Limitations of Sampling

- Current sampling methods are structure oblivious
  - But most queries are structure respecting!
- Most queries are actually range queries
  - "How much traffic from region X to region Y between 2am and 4am?"
- Much structure in data
  - Order (e.g. ordered timestamps, durations etc.)
  - Hierarchy (e.g. geographic and network hierarchies)
  - (Multidimensional) products of structures
- Can we make sampling structure-aware and improve accuracy?

Background on Sampling

- Inclusion Probability Proportional to Size (IPPS):
  - Given parameter τ, probability of sampling key with weight w is $\min\{1, w/\tau\}$
  - Key i has adjusted weight $a_i = w_i/p_\tau(w_i) = \max\{\tau, w_i\}$ (Horvitz-Thompson)
  - Can pick a τ so that expected sample size is k
- VarOpt sampling methods are Variance Optimal over keys:
  - Produces a sample of size exactly k keys using IPPS probabilities
  - Allow correlations between inclusion of keys (unlike Poisson sampling)
  - Give strong tail bounds on estimates via H-T estimates
  - But do not yet consider structure of keys

Probabilistic Aggregation

- We define a probabilistic aggregate of sampling probabilities:
  - Let vector $p \in [0,1]^n$ define sampling probabilities for n keys
  - Probabilistic aggregation to p’ sets entries to 0 or 1 so that:
    - $\forall$ i. $E[p'_i] = p_i$ (Agreement in expectation)
    - $\sum_i p'_i = \sum_i p_i$ (Agreement in sum)
    - $\forall$key sets J. $E[\prod_{i\in J} p'_i] \leq \prod_{i\in J} p_i$ (Inclusion bounds)
    - $\forall$key sets J. $E[\prod_{i\in J} (1-p'_i)] \leq \prod_{i\in J} (1-p_i)$ (Exclusion bounds)
- Apply probabilistic aggregation until all entries are set (0 or 1)
  - The 1 entries define the contents of the sample
  - This sample meets the requirements for a VarOpt sample

Pair Aggregation

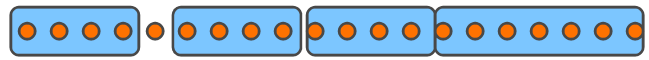
- Pair aggregation implements probabilistic aggregation
  - Pick two keys, i and j, such that neither is 0 or 1
  - If $p_i + p_j < 1$, one of them gets set to 0:
    - Pick j to set to 0 with probability $p_i/(p_i + p_j)$, or i with $p_j/(p_i + p_j)$
    - The other gets set to $p_i + p_j$ (preserving sum of probabilities)
  - If $p_i + p_j \geq 1$, one of them gets set to 1:
    - Pick i with probability $(1 - p_j)/(2 - p_i - p_j)$, or j with $(1 - p_i)/(2 - p_i - p_j)$
    - The other gets set to $p_i + p_j - 1$ (preserving sum of probabilities)
  - This satisfies all requirements of probabilistic aggregation
  - There is complete freedom to pick which pair to aggregate at each step
    - Use this to provide structure awareness by picking "close" pairs

Range Discrepancy

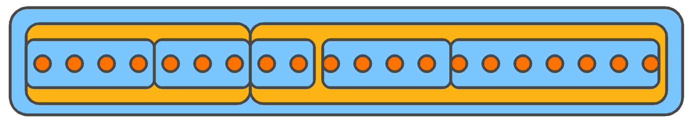
- We want to measure the quality of a sample on structured data
- Define range discrepancy based on difference between number of keys sampled in a range, and the expected number
  - Given a sample S, drawn according to a sample distribution p:
    Discrepancy of range R is $\Delta(S, R) = \mathrm{abs}(|S \cap R| - \sum_{i \in R} p_i)$
  - Maximum range discrepancy maximizes over ranges and samples:
    Discrepancy over sample dbn Ω is $\Delta = \max_{s \in \Omega} \max_{R \in \mathcal{R}} \Delta(S,R)$
  - Given range space $\mathcal{R}$, seek sampling schemes with small discrepancy

One-dimensional structures

- Can give very tight bounds for one-dimensional range structures
- $\mathcal{R}$ = Disjoint Ranges



  - Pair selection picks pairs where both keys are in same range R
  - Otherwise, pick any pair



- $\mathcal{R}$ = Hierarchy
  - Pair selection picks pairs with lowest LCA
- In both cases, for any $R \in \mathcal{R}$, $|S \cap R| \in \{\lfloor \sum_{i \in R} p_i \rfloor, \lceil \sum_{i \in R} p_i \rceil\}$
  - The maximum range discrepancy is optimal: $\Delta < 1$

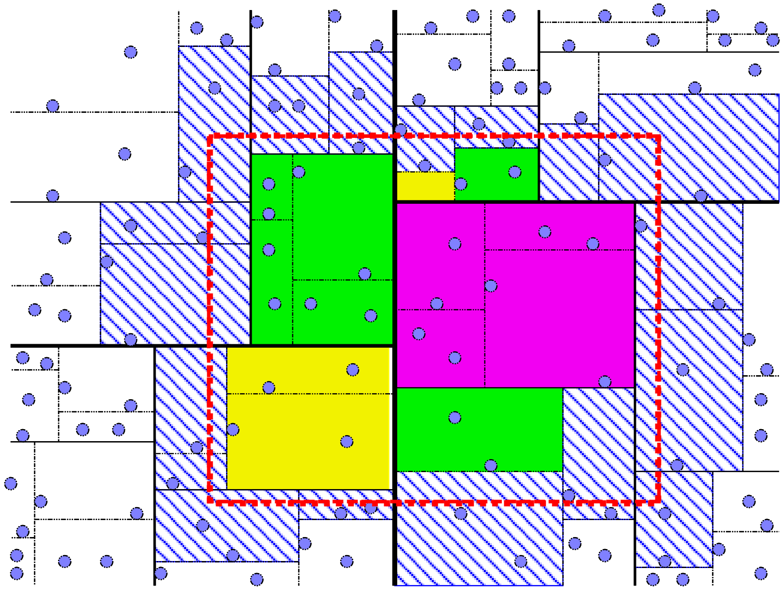
One-dimensional order

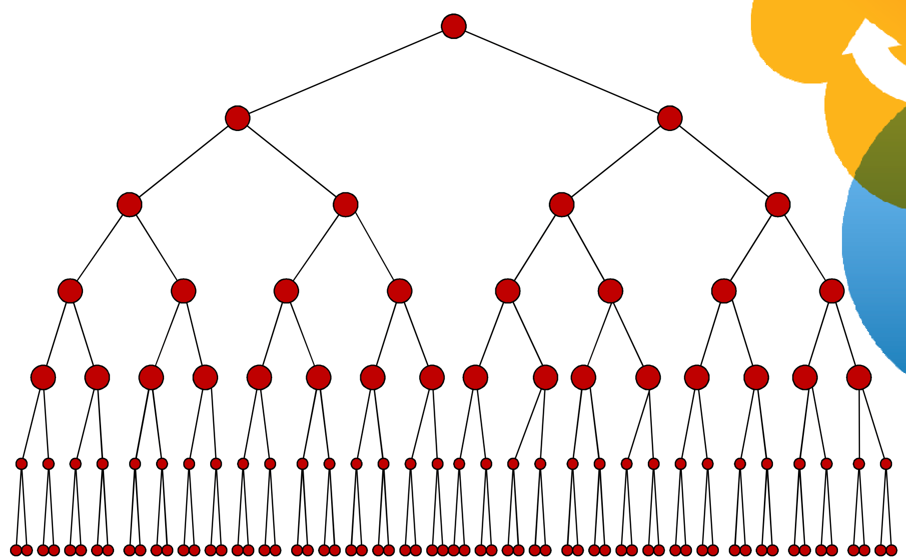
- $\mathcal{R}$ = order (i.e. points lie on a line in 1D)
  - Apply a left-to-right algorithm over the data in sorted order
  - For first two keys with $0 < p_i, p_j < 1$, apply pair aggregation
  - Remember which key was not set, find next unset key, pair aggregate
  - Continue right until all keys are set
- Sampling scheme for 1D order has discrepancy $\Delta < 2$
  - Analysis: view as a special case of hierarchy over all prefixes
  - Any $R \in \mathcal{R}$ is the difference of 2 prefixes, so has $\Delta < 2$
- This is tight: cannot give VarOpt distribution with $\Delta < 2$
  - For given Δ, we can construct a worst case input

Product Structures

- More generally, we have multidimensional keys
- E.g. (timestamp, bytes) is product of hierarchy with order
- KDHierarchy approach partitions space into regions
  - Make probability mass in each region approximately equal
  - Use KD-trees to do this. For each dimension in turn:
    - If it is an 'order' dimension, use median to split keys
    - If it is a 'hierarchy', find the split that minimizes the size difference
    - Recurse over left and right branches until we reach leaves

KD-Hierarchy Analysis





- Any query rectangle fully contains some rectangles, and cuts others
  - In d-dimensions on s leaves, at most $O(d\ s^{(d-1)/d} \log s)$ rectangles touched
  - Consequently, error is concentrated around $O((d \log^{1/2} s) s^{(d-1)/2d})$

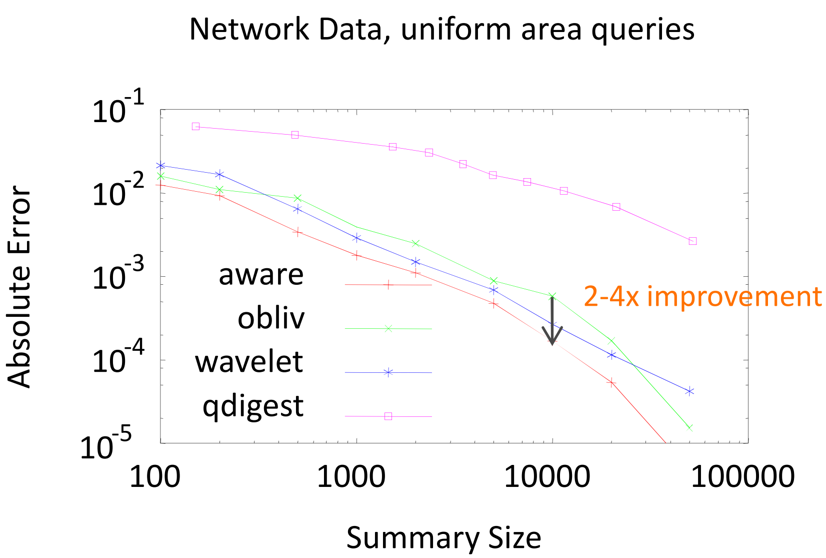
I/O efficient sampling for product spaces

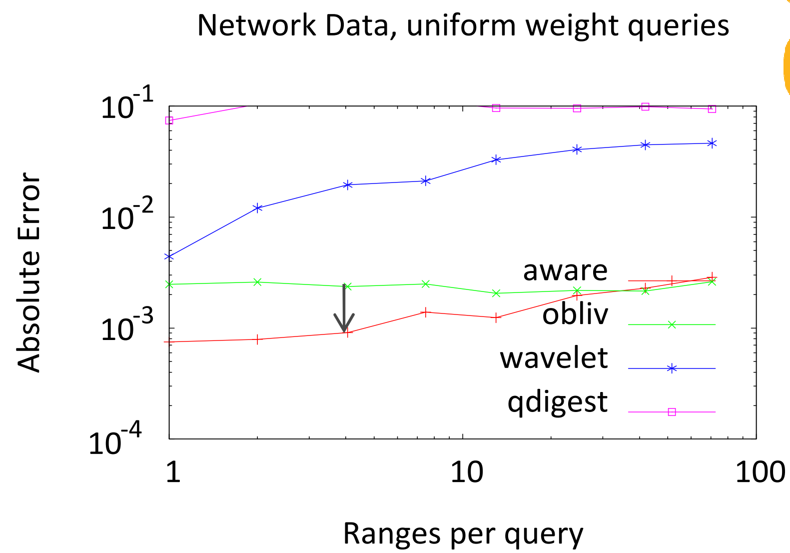
- Building the KD-tree over all data consumes a lot of space
- Instead, take two passes over data and use less space
  - Pass 1: Compute uniform sample of size s' > s and build tree
  - Pass 2: Maintain one key for each node in the tree
    - When two keys fall in same node, use pair aggregation
    - At end, pair aggregate up the binary tree to generate final sample
    - Conclude with a sample of size s, guided by structure of tree
- Variations of the same approach work for 1D structures

Experimental Study

- Compared structure aware I/O Efficient Sampling to:
  - VarOpt 'obliv' (structure unaware) sampling
  - Qdigest: Deterministic summary for range queries
  - Sketches: Randomized summary based on hashing
  - Wavelets: 2D Haar wavelets – generate all coefficients, then prune
- Studied on various data sets with different size, structure
  - Shown here: network traffic data (product of 2 hierarchies: $2^{32} \times 2^{32}$)
  - Query loads: uniform area rectangles, and uniform weight rectangles

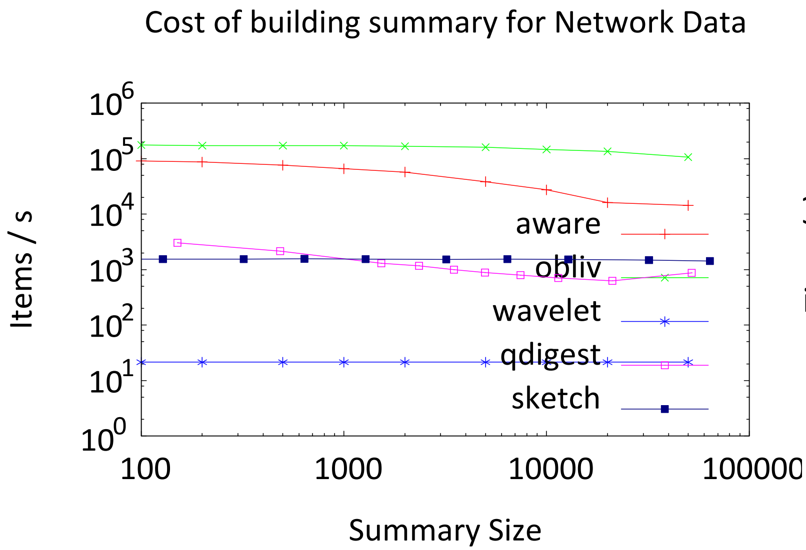
Accuracy results

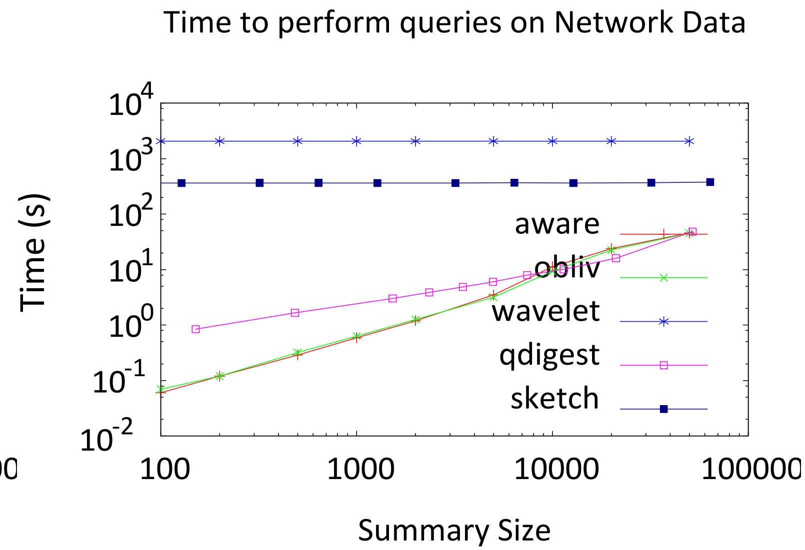




- Compared on uniform area queries, and uniform weight queries
- Clear benefit to structure aware sampling
- Wavelet sometimes competitive but very slow

Scalability Results





- Structure aware sampling is somewhat slower than VarOpt
  - But still much faster than everything else, particularly wavelets
- Queries take same time to perform for both sampling methods
  - Just answer query over the sample

Concluding Remarks

- Structure aware sampling can improve accuracy greatly
  - For structure-respecting queries
  - Result is still variance optimal
- The streaming (one-pass) case is harder
  - There is a unique VarOpt sampling distribution
  - Instead, must relax VarOpt requirement
  - Initial results in SIGMETRICS'11