Algorithms for Querying Noisy Distributed/Streaming Datasets

Qin Zhang

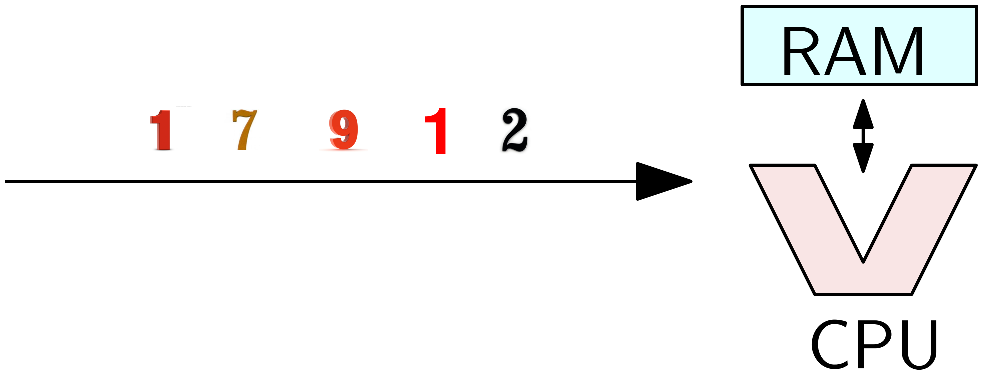
Indiana University Bloomington

Sublinear Algo Workshop @ JHU

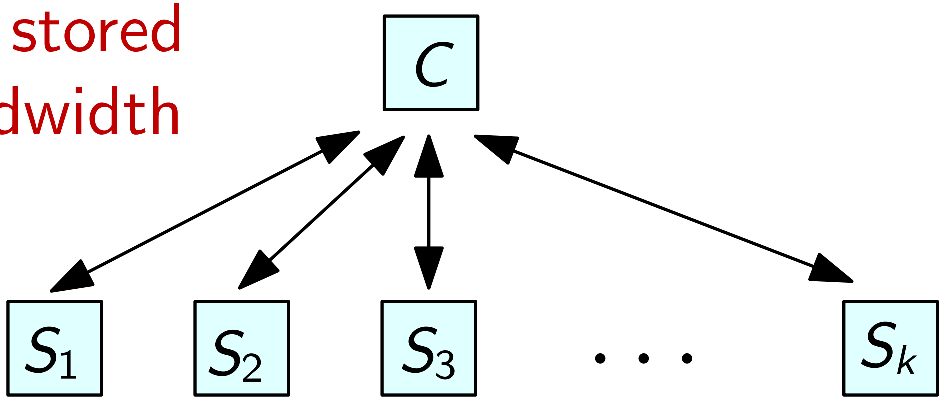
Jan 9, 2016

The “big data” models

- The **streaming** model (Alon, Matias and Szegedy 1996)
  – *high-speed* online data
  – *limited* storage



- The **k-site** model
  – data is *distributedly* stored
  – *limited* network bandwidth



k-site model

k **sites and** 1 **coordinator**.

– each site has a 2-way communication channel with the coordinator.

– each site S_i has a piece of data x_i. The coordinator has $\emptyset$.

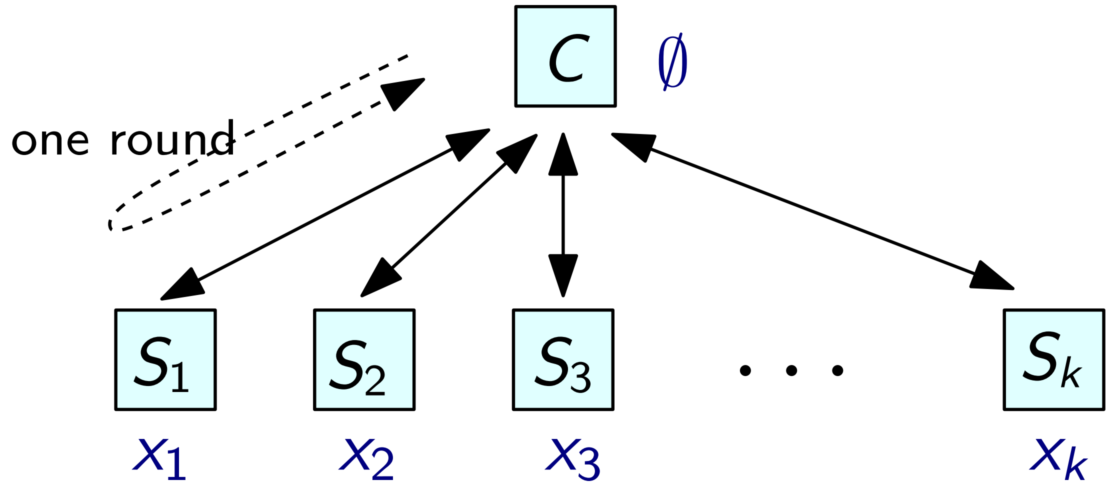
Task: compute $f(x_1, \ldots, x_k)$ together via communication.

– The coordinator reports the answer.

– computation is divided into rounds.

Goal: minimize both

- total #bits of comm. (o(Input); best polylog(Input))
- and #rounds ($O(1)$ or polylog(Input)).



k-site model

k **sites and** 1 **coordinator**.

– each site has a 2-way communication channel with the coordinator.

– each site S_i has a piece of data x_i. The coordinator has $\emptyset$.

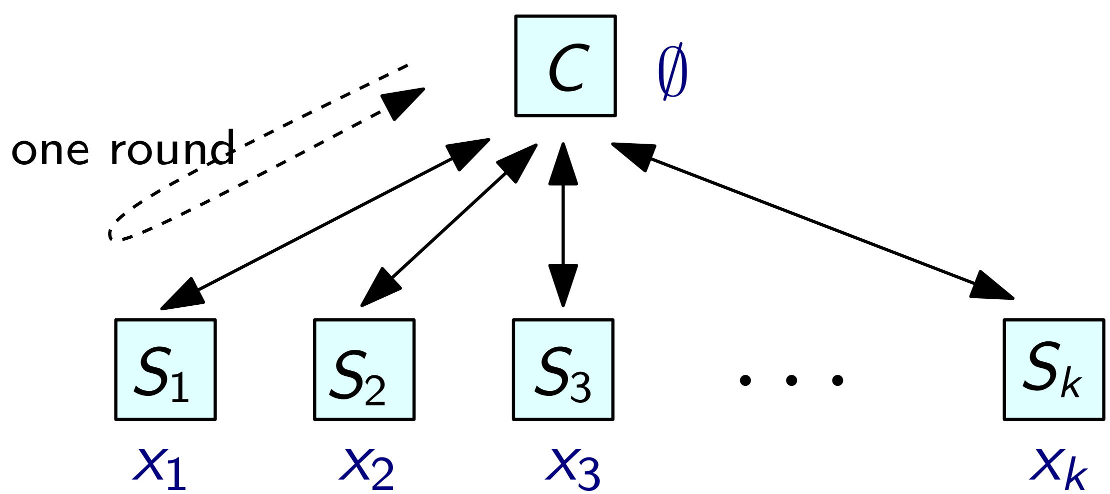
Task: compute $f(x_1, \ldots, x_k)$ together via communication.

– The coordinator reports the answer.

– computation is divided into rounds.

Goal: minimize both

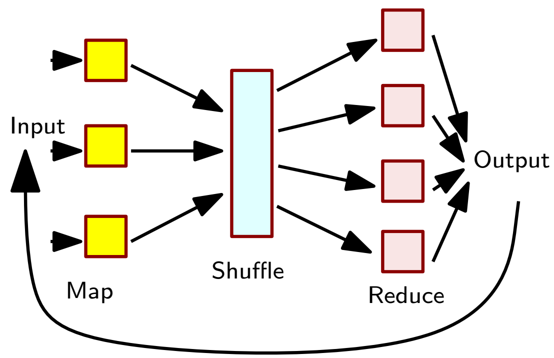
- total #bits of comm. (o(Input); best polylog(Input))
- and #rounds ($O(1)$ or polylog(Input)).



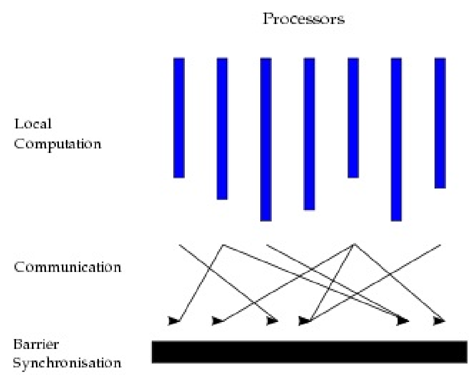
– no constraint on #bits can be sent or received by each site at each round. (usually balanced)
– do not count local computation (usually linear)

k-site model (cont.)

Communication $\rightarrow$ time, energy, bandwidth, ...

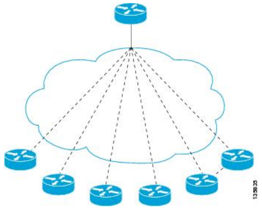


The MapReduce model.



The BSP model.

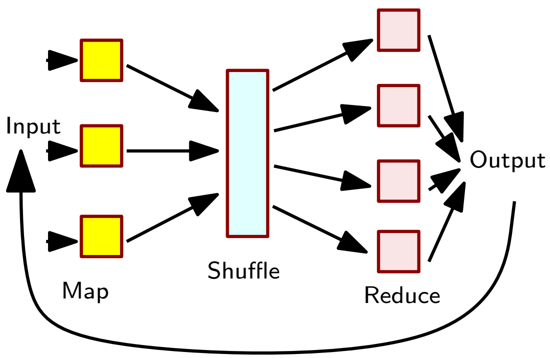
Also network monitoring, sensor networks, etc.



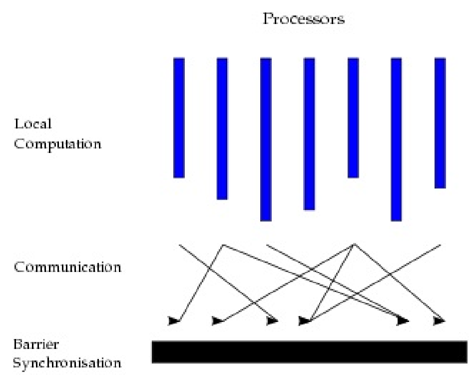
Abstraction

k-site model (cont.)

Communication $\rightarrow$ time, energy, bandwidth, . . .

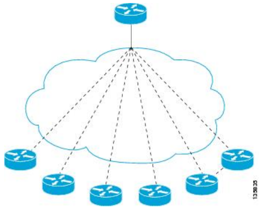


The MapReduce model.



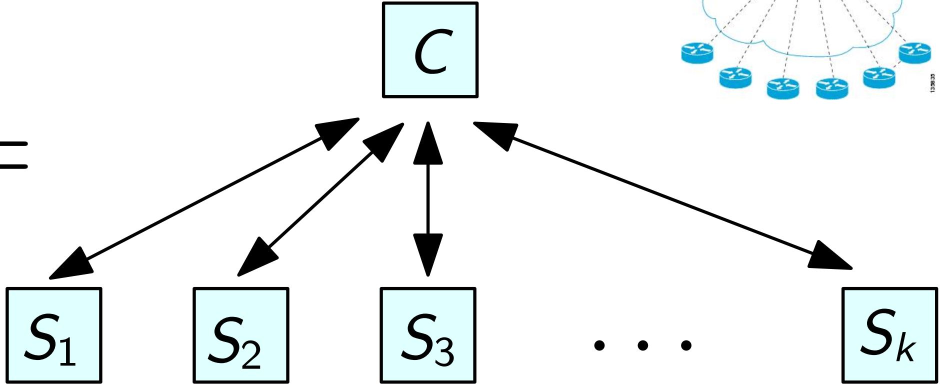
The BSP model.

Also network monitoring, sensor networks, etc.



Abstraction

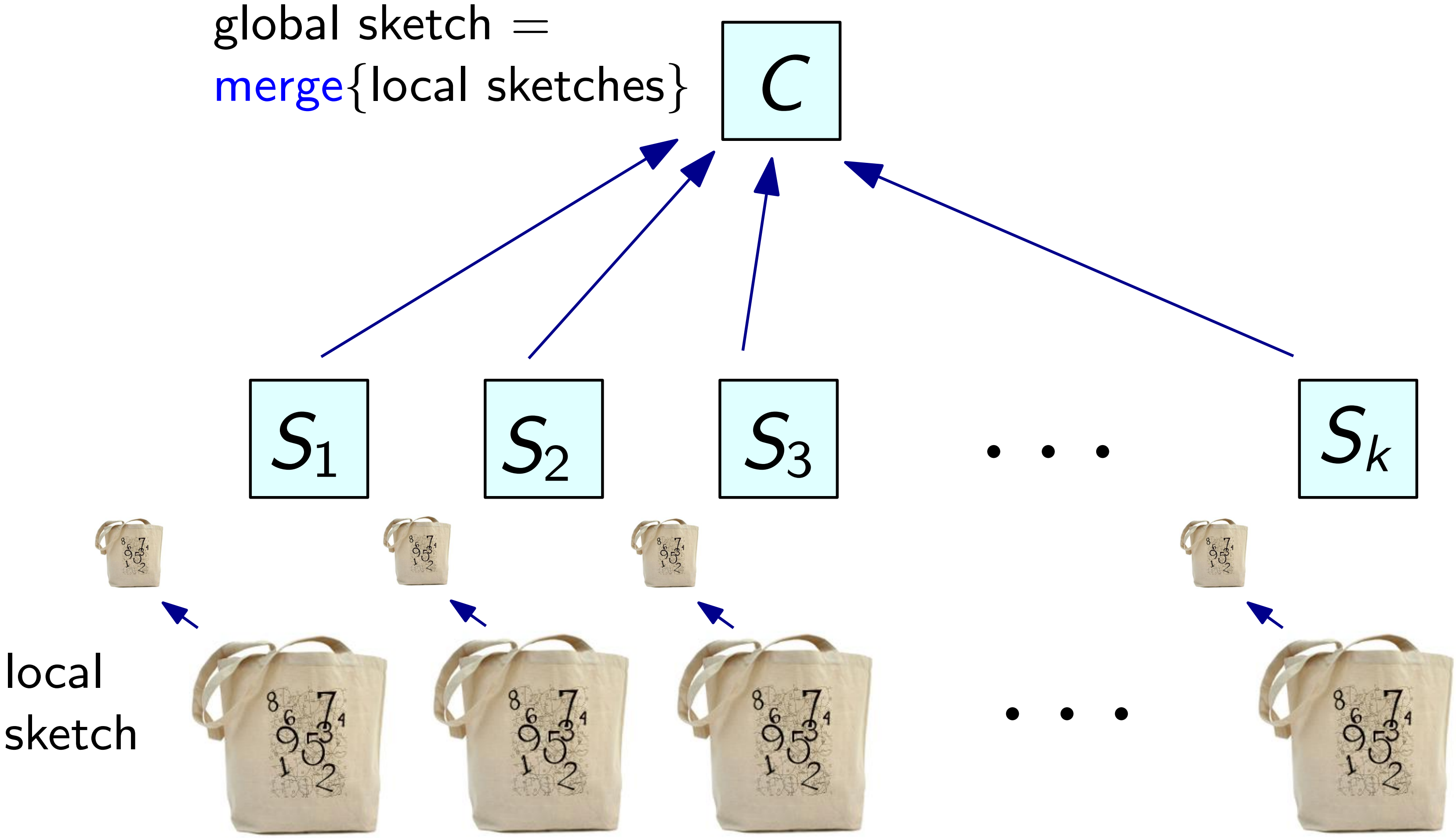
=



We will start with the k-site model, and will mention the streaming model at the end

Sketching

Q: How many distinct elements (F_0) in the **union** of the k bags?



Linear sketching

- **Random linear mapping** $M : R^n \to R^k$ where $k \ll n$.

$$\begin{bmatrix} \quad M \quad \end{bmatrix} \begin{bmatrix} x \\ \\ \\ \end{bmatrix} = \begin{bmatrix} Mx \end{bmatrix} \longrightarrow g(Mx) \approx f(x)$$

linear mapping

sketching vector

The data. e.g.,
a frequency vector

Linear sketching

- **Random linear mapping** $M : R^n \to R^k$ where $k \ll n$.

$$\underbrace{\begin{bmatrix} \quad & M & \quad \end{bmatrix}}_{\text{linear mapping}} \begin{bmatrix} \\ x \\ \\ \\ \end{bmatrix} = \underbrace{\begin{bmatrix} Mx \end{bmatrix}}_{\text{sketching vector}} \longrightarrow g(Mx) \approx f(x)$$

The data. e.g.,
a frequency vector

- **Perfect** for distributed and streaming computation

Linear sketching

- **Random linear mapping** $M : R^n \to R^k$ where $k \ll n$.

$$\begin{bmatrix} \quad & M & \quad \end{bmatrix} \begin{bmatrix} \\ x \\ \\ \end{bmatrix} = \begin{bmatrix} Mx \end{bmatrix} \longrightarrow g(Mx) \approx f(x)$$

linear mapping

sketching vector

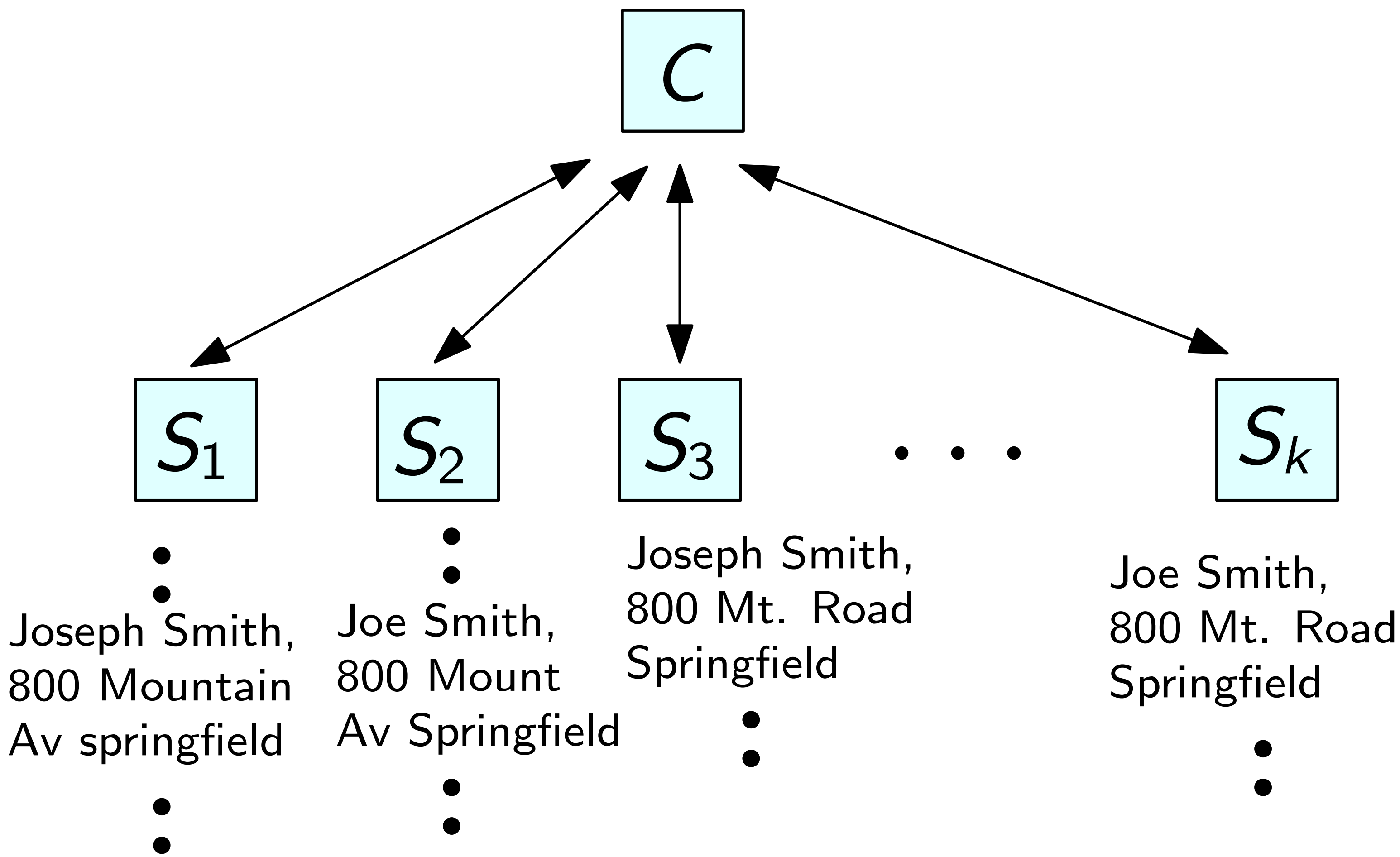
The data. e.g.,
a frequency vector

- **Perfect** for distributed and streaming computation

- **Simple and useful**: used in many statistical/graph/algebraic problems in streaming, compressive sensing, ...

But what if the data is noisy?

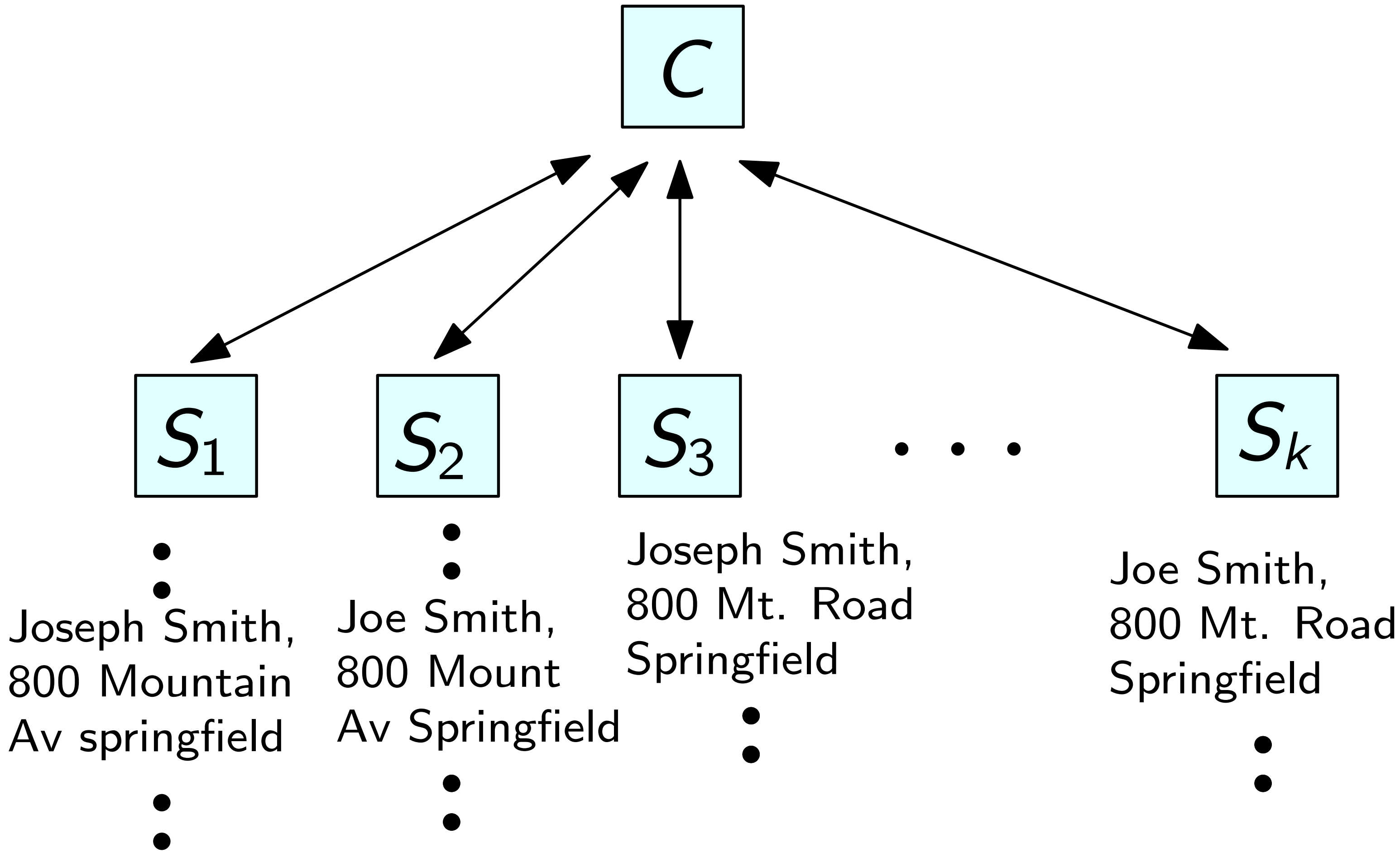
Real world distributed datasets are often noisy!



But what if the data is noisy?

Real world distributed datasets are often noisy!

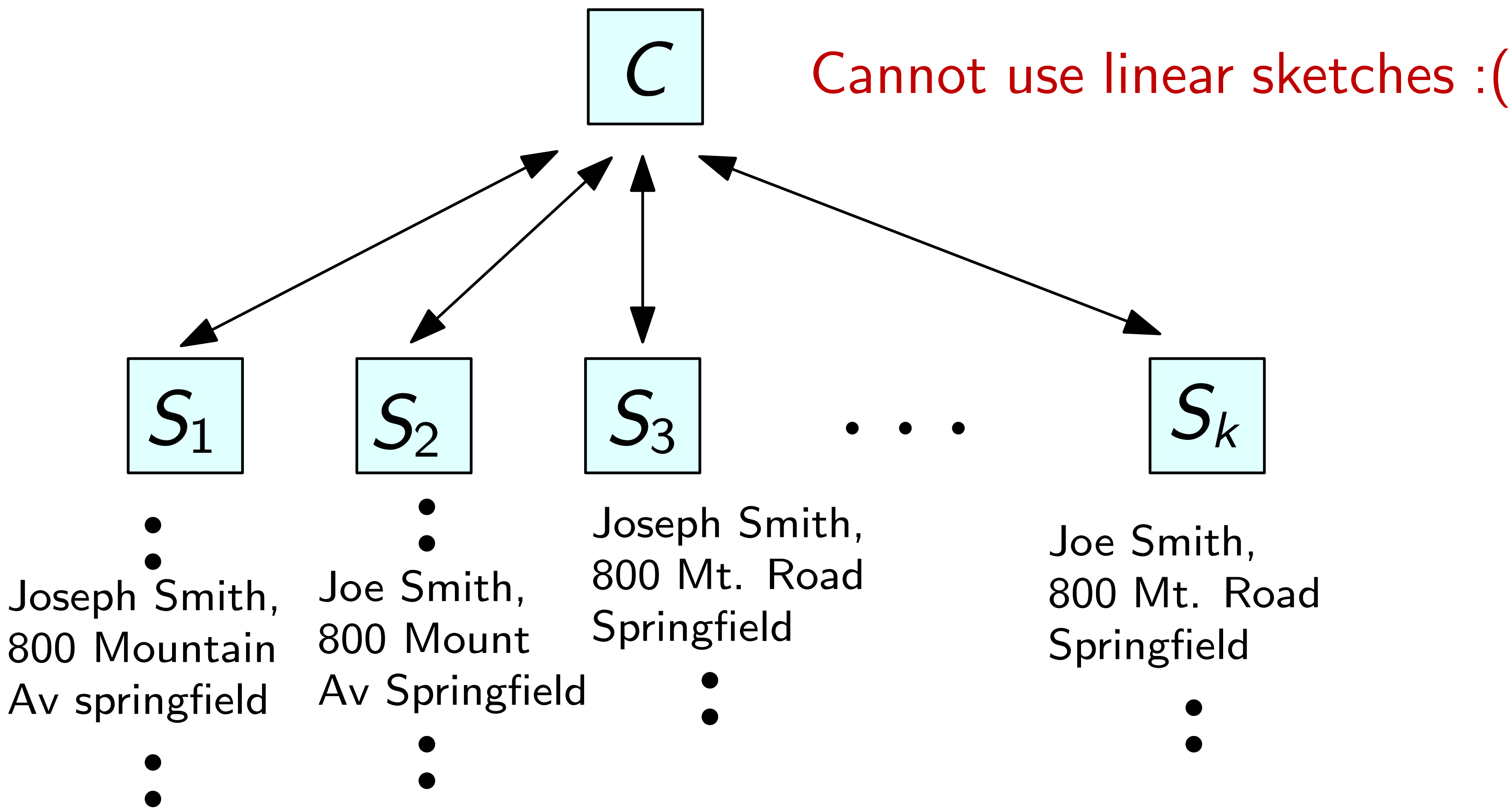
We (have to) consider similar items as one element. Then how to compute F_0?



But what if the data is noisy?

Real world distributed datasets are often noisy!

We (have to) consider similar items as one element. Then how to compute F_0?



Noisy data is universal

- Music, Images, ...
  After compressions, resize,
  reformat, etc.

Noisy data is universal

- 

  Music, Images, ...
  After compressions, resize, reformat, etc.

- 

  “sublinear algorithm workshop 2016”
  “JHU sublinear algorithm”
  “sublinear John Hopkins”

Queries of the same meaning sent to Google

Related to Entity Resolution

- Related to Entity Resolution: Identify and link/group different manifestations of the same real world object.

  Very important in data cleaning / integration. Have been studied for 40 years in DB, also in AI, NT.

  E.g. [Gill& Goldacre'03, Koudas et al.'06, Elmagarmid et al.'07, Herzog et al.'07, Dong& Naumann'09, Willinger et al.'09, Christen'12] for introductions, and [Getoor and Machanavajjhala'12] for a toturial.

  Centralized, detect items representing the same entity, merge/output all distinct entities.

Related to Entity Resolution

- Related to Entity Resolution: Identify and link/group different manifestations of the same real world object.

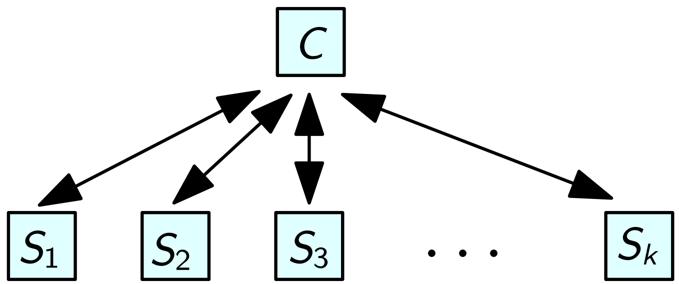
  Very important in data cleaning / integration. Have been studied for 40 years in DB, also in AI, NT.

  E.g. [Gill& Goldacre'03, Koudas et al.'06, Elmagarmid et al.'07, Herzog et al.'07, Dong& Naumann'09, Willinger et al.'09, Christen'12] for introductions, and [Getoor and Machanavajjhala'12] for a toturial.

  Centralized, detect items representing the same entity, merge/output all distinct entities.

- In the big data models, we want communication/space-efficient algorithms (o(input size)); cannot afford a comprehensive de-duplication.

Our problems and goal

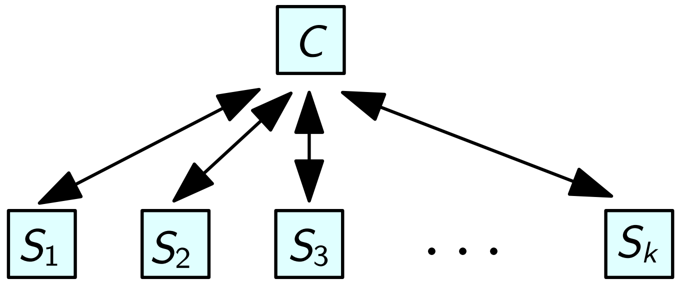


Problem: how to perform in the k-site model robust statistical estimation comm. efficiently?

Assume all parties are provided with an oracle (e.g., a distance function and a threshold) determining whether two items u, v rep. the same entity (denoted by $u \sim v$) or not

We will design a framework so that users can plug-in any "distance function" at run time.

Our problems and goal



Problem: how to perform in the k-site model robust statistical estimation comm. efficiently?

Assume all parties are provided with an oracle (e.g., a distance function and a threshold) determining whether two items u, v rep. the same entity (denoted by $u \sim v$) or not

We will design a framework so that users can plug-in any "distance function" at run time.

Goal: minimize communication & #rounds

Remarks

Remark 1. We do not specify the distance function in our algorithms, for two reasons:

(1) Allows our algorithms to work with *any* distance functions.

(2) Sometimes it is very hard to assume that similarities between items can be expressed by a well-known distance function: "AT&T Corporation" is closer to "IBM Corporation" than "AT&T Corp" under the edit distance!

Remarks

Remark 1. We do not specify the distance function in our algorithms, for two reasons:

(1) Allows our algorithms to work with *any* distance functions.

(2) Sometimes it is very hard to assume that similarities between items can be expressed by a well-known distance function: "AT&T Corporation" is closer to "IBM Corporation" than "AT&T Corp" under the edit distance!

Remark 2. We assume transitivity: if $u \sim v$, $v \sim w$ then $u \sim w$. In other words, the noise is "well-shaped".

One may come up with the following problematic situation: we have $a \sim b$, $b \sim c$, $\ldots$, $y \sim z$, however, $a \not\sim z$.

For many specific metic spaces, our algorithms still work if the number of "outliers" is small.

Remarks (cont.)

Remark 3. Clustering will help?

Answer: NO. #clusters can be linear.

Remarks (cont.)

Remark 3. Clustering will help?

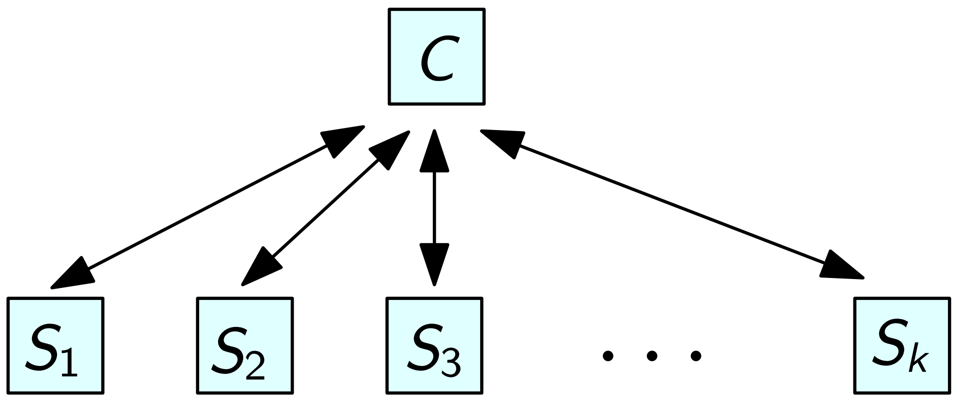
Answer: NO. #clusters can be linear.

Remark 4. Does there exist a magic hash function that
(1) map (only) items in same group into same bucket
and
(2) can be described succinctly?

Answer: NO

For specific metrics, tools such as LSHs may help

A few notations



- We have k sites (machines), each holding a multiset of items S_i.
- Let multiset $S = \bigcup_{i \in [k]} S_i$, let $m = |S|$.
- Under the transitivity assumption, S can be partitioned into a set of groups $\mathcal{G} = \{G_1, \ldots, G_n\}$. Each group G_i represents a distinct universe element.
- $\tilde{O}(\cdot)$ hides $\text{poly}\log(m/\epsilon)$ factors.

Our results

	noisy data		noise-free data
	(comm.) items	rounds	bits
F_0	$\tilde{O}(\min\{k/\epsilon^3, k^2/\epsilon^2\})$	$\tilde{O}(1)$	$\Omega(k/\epsilon^2)$ [WZ12,WZ14]
L_0-sampling	$\tilde{O}(k)$	$\tilde{O}(1)$	$\Omega(k)$
F_p $(p \geq 1)$	$\tilde{O}((k^{p-1} + k^3)/\epsilon^3)$	$O(1)$	$\Omega(k^{p-1}/\epsilon^2)$ [WZ12]
(ϕ, ϵ)-HH	$\tilde{O}(\min\{k/\epsilon, 1/\epsilon^2\})$	$O(1)$	$\Omega(\min\{\frac{\sqrt{k}}{\epsilon}, \frac{1}{\epsilon^2}\})$ [HYZ12,WZ12]
Entropy	$\tilde{O}(k/\epsilon^2)$	$O(1)$	$\Omega(k/\epsilon^2)$ [WZ12]

1. *p-th frequency moment* $F_p(S) = \sum_{i \in [n]} |G_i|^p$.

   We consider F_0 and F_p $(p \geq 1)$, and allow a $(1+\epsilon)$-approximation.

2. *L_0-sampling* on S: return a group G_i (or an arbitrary item in G_i) uniformly at random from $\mathcal{G}$.

3. *(ϕ, ϵ)-heavy-hitter* of S $(0 < \epsilon \leq \phi \leq 1)$ (definition omitted)

4. *Empirical entropy*: $\text{Entropy}(S) = \sum_{i \in [n]} \frac{|G_i|}{m} \log \frac{m}{|G_i|}$.

   We allow a $(1+\epsilon)$-approximation.

Take-home message:

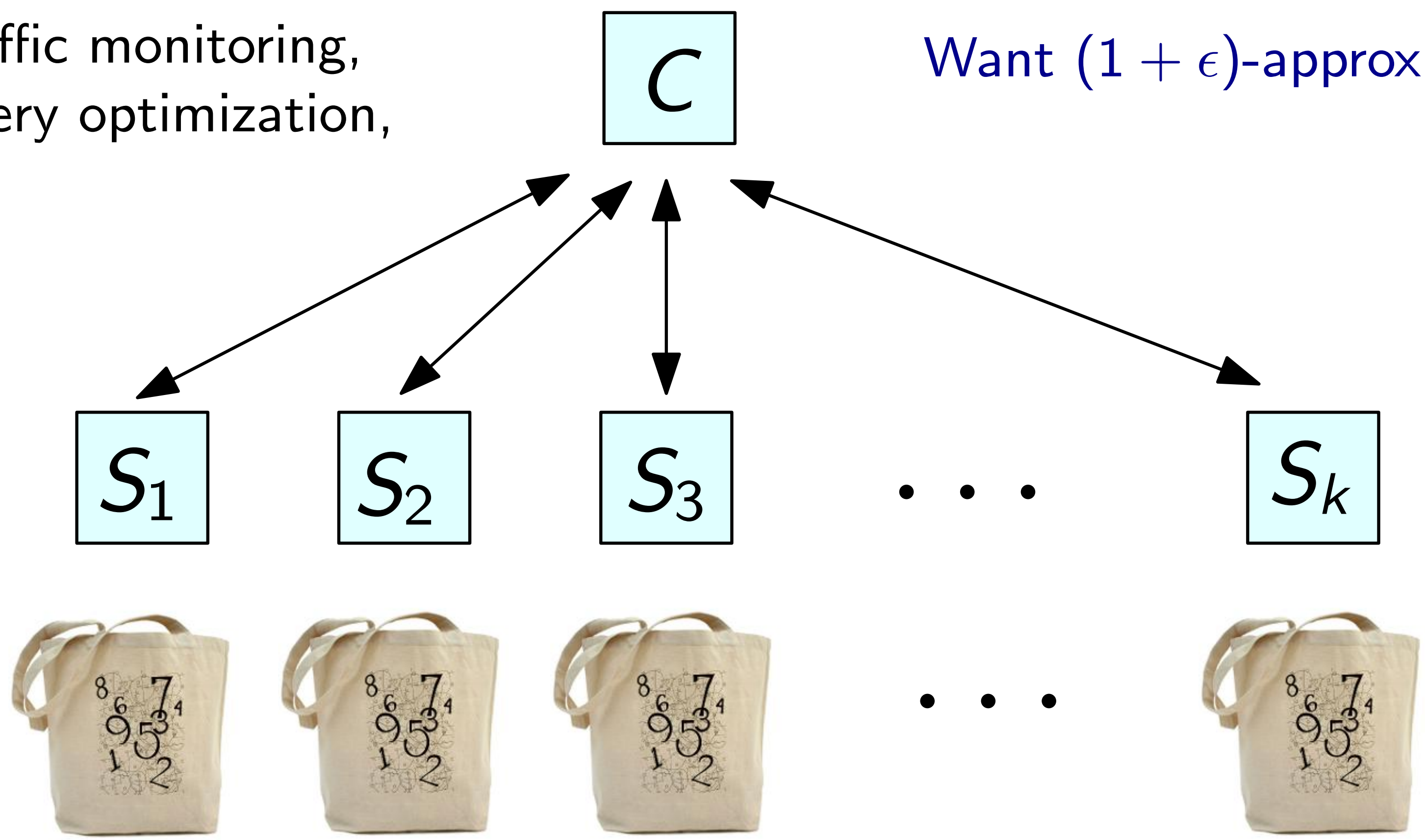
In the distributed setting, we can handle well-shaped noise in several statistical estimations almost for free in terms of communication

Rest of the talk: Algorithms for F_0

Q: How many distinct elements/groups in the **union** of the k bags?

Important in:
traffic monitoring,
query optimization,
...

Want $(1+\epsilon)$-approximation



1. Simple-Sampling

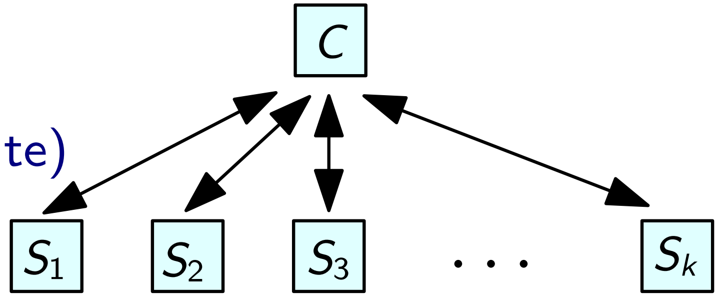
Simple.
$\tilde{O}(k^2/\epsilon^2)$ comm. 2 rounds.

2. Advanced-Sampling

A bit more complicated.
$\tilde{O}(k/\epsilon^3)$ comm. $\tilde{O}(1)$ rounds

Better than $\tilde{O}(k^2/\epsilon^2)$ bits in the sense that
(1) we want to scale on k
(2) used in the algo for ℓ_0-sampling with $\epsilon = \Theta(1)$

Simple-Sampling



Algorithm Simple-Sampling

(assuming local de-duplication is done at each site)

1. Let $m = |S| = \sum_{i \in [k]} |S_i|$.
2. For $j = 1, \ldots, \eta = \Theta(k/\epsilon^2)$
   (a) jointly sample a random item $u_j \in S$;
   Let G_{u_j} be the group containing u_j.
   (b) jointly compute $\left|G_{u_j}\right|$, and set $X_j = 1/\left|G_{u_j}\right|$.
3. Output $\frac{m}{\eta}\sum_{j \in [k]} X_j$.

Theorem

Simple-Sampling gives a $(1+\epsilon)$-approximation of F_0 with probability $2/3$ using $\tilde{O}(k^2/\epsilon^2)$ bits and 2 rounds.

Advanced-Sampling

Main idea: reduce the variance of X_j in Simple-Sampling

– If we can partition all groups in $\mathcal{G}$ into classes $\mathcal{G}_0, \ldots, \mathcal{G}_{\log k}$ such that $\mathcal{G}_\ell = \{G \in \mathcal{G} \mid |G| \in (2^{\ell-1}, 2^\ell]\}$, and run Algo Simple-Sampling on each class individually, we can shave a factor of k in the number of samples X_j needed ($\eta : k/\epsilon^2 \to 1/\epsilon^2$).

Advanced-Sampling

Main idea: reduce the variance of X_j in Simple-Sampling

– If we can partition all groups in $\mathcal{G}$ into classes $\mathcal{G}_0, \ldots, \mathcal{G}_{\log k}$ such that $\mathcal{G}_\ell = \{G \in \mathcal{G} \mid |G| \in (2^{\ell-1}, 2^\ell]\}$, and run Algo Simple-Sampling on each class individually, we can shave a factor of k in the number of samples X_j needed ($\eta : k/\epsilon^2 \to 1/\epsilon^2$).

– However, we cannot afford to partition the groups into classes in the distributed setting.

Advanced-Sampling

Main idea: reduce the variance of X_j in Simple-Sampling

– If we can partition all groups in $\mathcal{G}$ into classes $\mathcal{G}_0, \ldots, \mathcal{G}_{\log k}$ such that $\mathcal{G}_\ell = \{G \in \mathcal{G} \mid |G| \in (2^{\ell-1}, 2^\ell]\}$, and run Algo Simple-Sampling on each class individually, we can shave a factor of k in the number of samples X_j needed ($\eta : k/\epsilon^2 \rightarrow 1/\epsilon^2$).

– However, we cannot afford to partition the groups into classes in the distributed setting.

Our techniques:

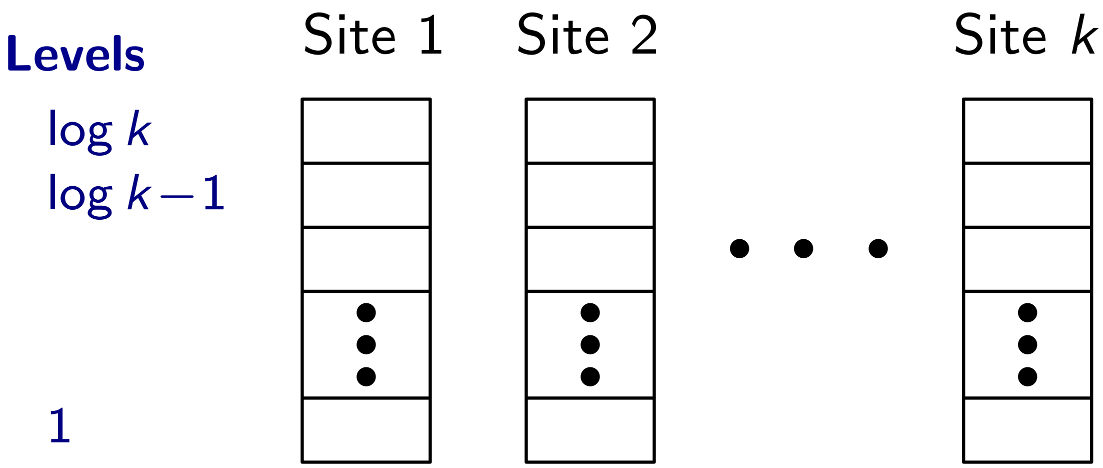
local hierarchical partition

$+$ distributed rejection sampling

Advanced-Sampling (cont.)

Our techniques:

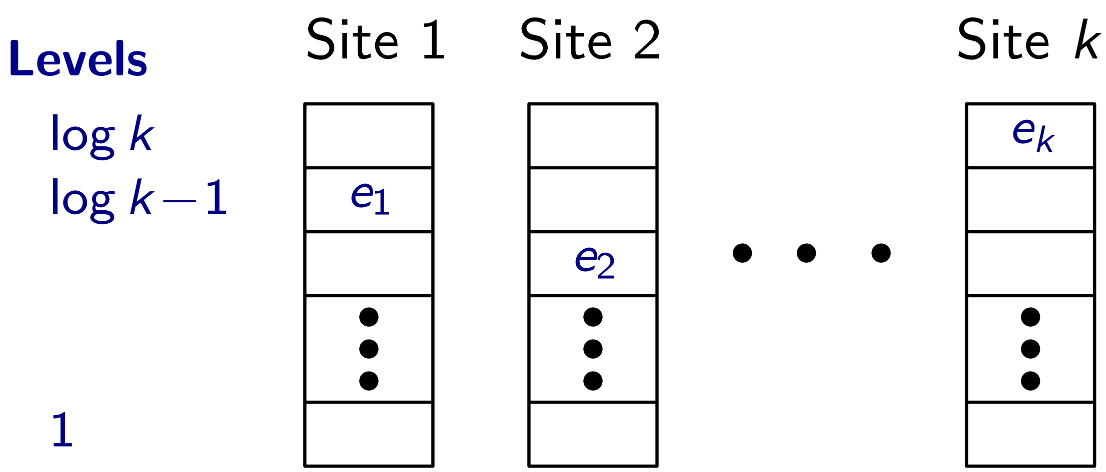
Local hierarchical partition: at site i about $|S_i|/2^\ell$ at level ℓ.



Advanced-Sampling (cont.)

Our techniques:

Local hierarchical partition: at site i about $|S_i|/2^\ell$ at level ℓ.



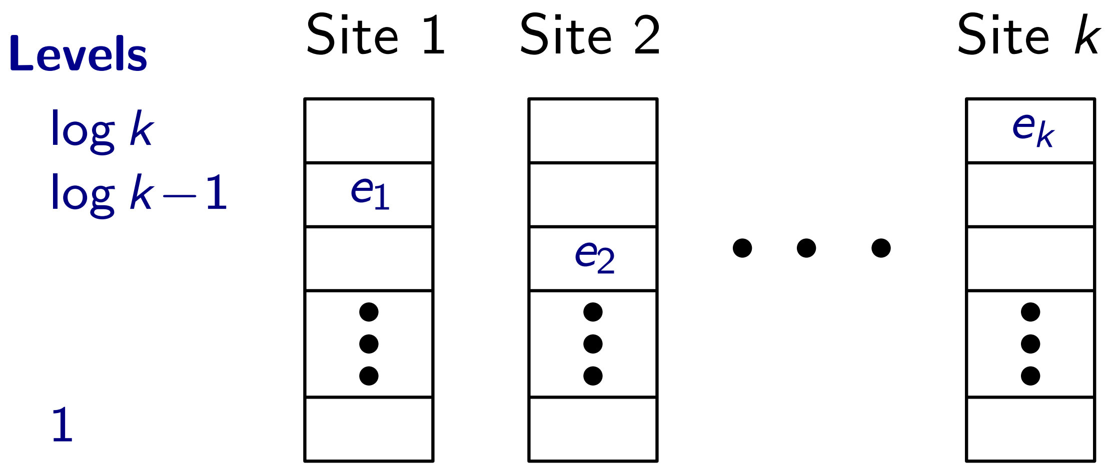
$e_1, e_2, \ldots, e_k \in G$

Have inconsistency, $u \sim v$ but u, v are sampled at different levels at different sites.

Advanced-Sampling (cont.)

Our techniques:

Local hierarchical partition: at site i about $|S_i|/2^\ell$ at level ℓ.



$e_1, e_2, \ldots, e_k \in G$

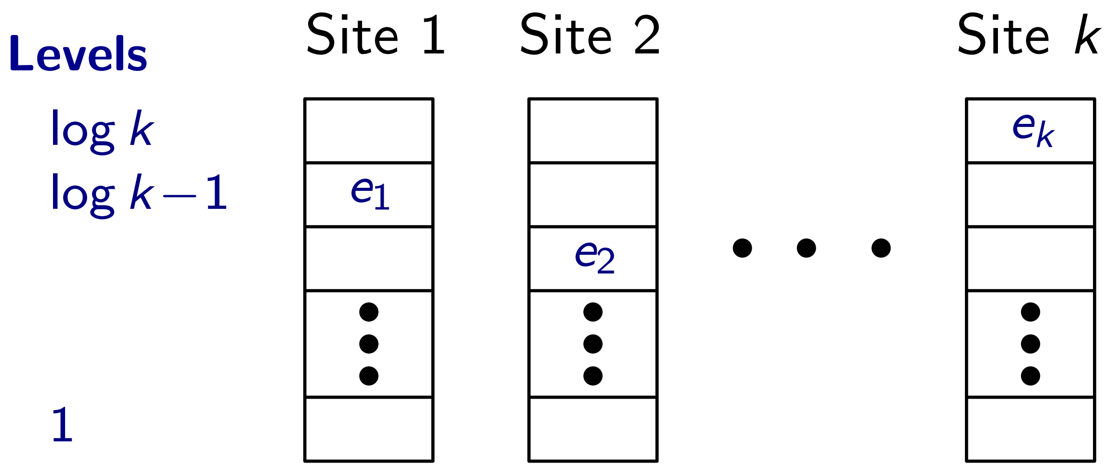
$level(G) = \max_i level(e_i)$

Have inconsistency, $u \sim v$ but u, v are sampled at different levels at different sites.

Advanced-Sampling (cont.)

Our techniques:

Local hierarchical partition: at site i about $|S_i|/2^{\ell}$ at level ℓ.



$e_1, e_2, \ldots, e_k \in G$

$level(G) = \max_i level(e_i)$

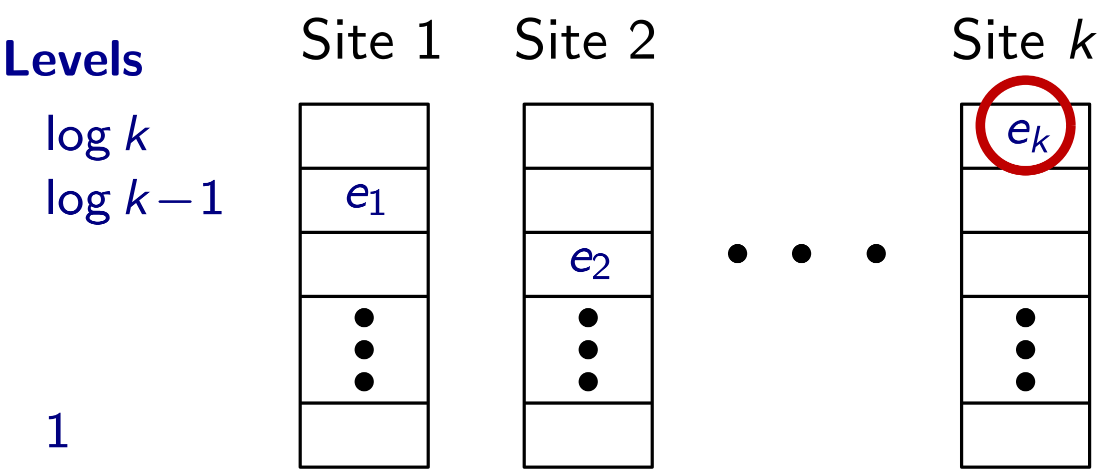
Note: $level(G) \neq class(G)$ but close :)

Have inconsistency, $u \sim v$ but u, v are sampled at different levels at different sites.

Advanced-Sampling (cont.)

Our techniques:

Local hierarchical partition: at site i about $|S_i|/2^\ell$ at level ℓ.



$e_1, e_2, \ldots, e_k \in G$

$level(G) = \max_i level(e_i)$

Note: $level(G) \neq class(G)$ but close :)

Have inconsistency, $u \sim v$ but u, v are sampled at different levels at different sites.

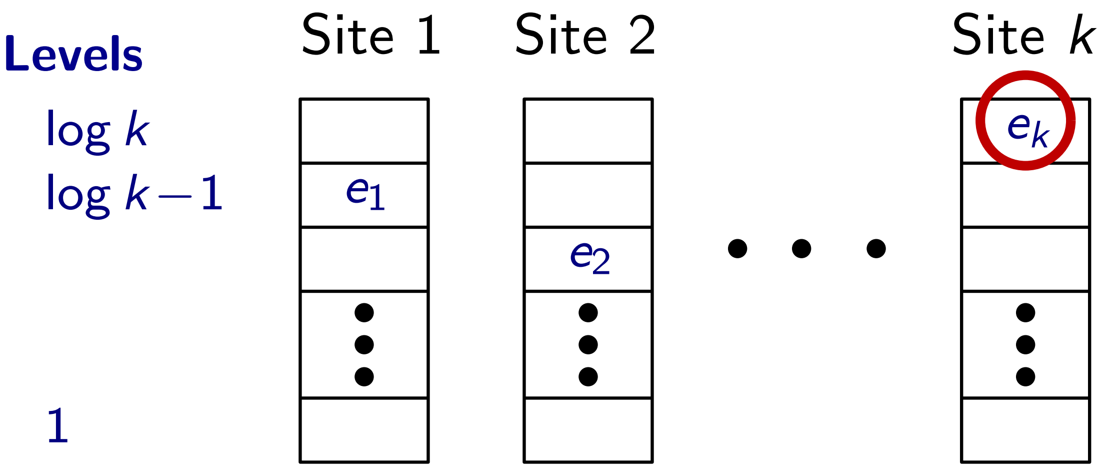
$+$ Distributed rejection sampling: resolve the inconsistency

The k sites jointly sample items as before, but only for those items e with $level(e) = level(G_e)$ (how?), compute $1/w(G_e)$ as X_j

Advanced-Sampling (cont.)

Our techniques:

Local hierarchical partition: at site i about $|S_i|/2^\ell$ at level ℓ.



$e_1, e_2, \ldots, e_k \in G$

$level(G) = \max_i level(e_i)$

Note: $level(G) \neq class(G)$ but close :)

Have inconsistency, $u \sim v$ but u, v are sampled at different levels at different sites.

$+$ Distributed rejection sampling: resolve the inconsistency

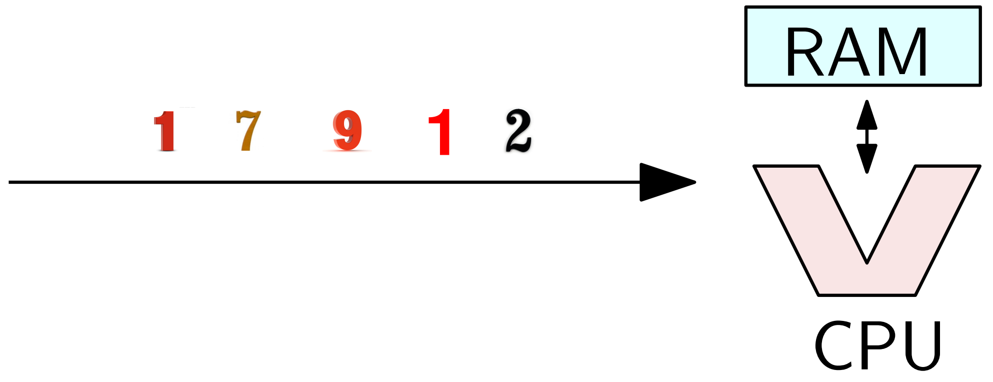
The k sites jointly sample items as before, but only for those items e with $level(e) = level(G_e)$ (how?), compute $1/w(G_e)$ as X_j

Repeat until we get $\tilde{O}(1/\epsilon^2)$ X_j's for **each** level of groups, and then run the estimation of *Simple-Sampling* for each level.

Other problems

1. L_0-**sampling**: $\tilde{O}(k)$ communication and $\tilde{O}(1)$ rounds.
   – Use the algorithm for F_0 as a subroutine
2. p-**th frequency moment**: $\tilde{O}((k^{p-1} + k^3)/\epsilon^3)$ comm. and $\tilde{O}(1)$ rounds.
   – Adapt an algo by Kannan, Vempala and Woodruff. (COLT 2014)
3. (ϕ, ϵ)-**heavy-hitter**: $\tilde{O}(\min\{k/\epsilon, 1/\epsilon^2\})$ comm. and $O(1)$ rounds.
   – Easy
4. **Empirical entropy**: $\tilde{O}(k/\epsilon^2)$ comm. and $O(1)$ rounds.
   – Adapt an algo by Chakrabarti, Cormode and McGregor (SODA 2007) in streaming

Now a bit on the streaming model



The streaming model

Q: Can we adapt the algorithms for the k-site model to the streaming model?

– the *simple-sampling* needs to revisit the data (2 rounds)

– the *advanced-sampling* needs more rounds

Not sure if we can do it for general metric spaces.

Can do for some specific metric spaces. For example, for $O(1)$-Euclidean space and well-shaped datasets, there exists a streaming algo using space $\tilde{O}(1/\epsilon^2)$ (Chen, Z., 2016).

Experiments (streaming model)

- **Problem**: compute the number of robust distinct elements (F_0) in the streaming model

  Given a threshold α, partition items in the input set S to a *minimum* set of groups $\mathcal{G} = \{G_1, \ldots, G_n\}$ so that $\forall p, q \in G_i, d(p, q) \leq \alpha$.

- **Data**: $4,000,000$ images from `ImageNet`, converted into points in the Euclidean space

- **Computing environment**: a desktop PC with 8GB of RAM and a 4-core 3.40GHz Intel i7 CPU

Experiments (known α)

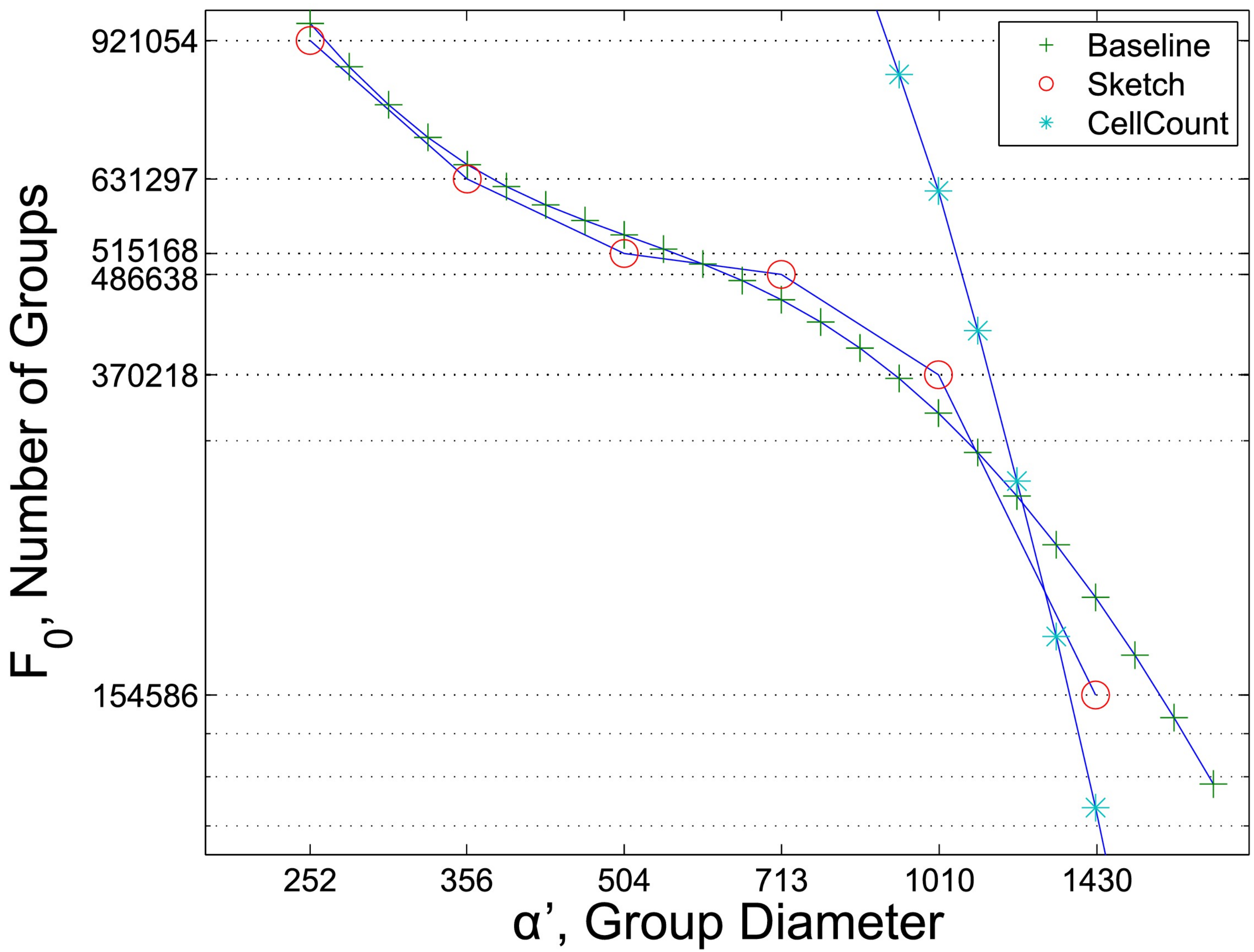
No. pts	9,000	18,000	36,000	72,000
`I500k100x5d`	22.8%	10.6%	8.3%	6.6%
`I500k10x5d`	15.8%	9.2%	6.7%	5.7%
`I500k2x5d`	5.2%	3.0%	2.8%	2.2%
`I4m2x5d`	6.0%	3.5%	3.3%	2.4%

Table 6: Vary duplication ratio; average error over 20 runs; median output of 6 sketches; known α.

No. pts	9,000	18,000	36,000	72,000	144,000
`I4m2x5d`	6.0%	3.5%	3.3%	2.4%	1.7%
`I4m2x10d`	5.8%	4.2%	3.4%	2.6%	1.5%
`I4m2x20d`	6.4%	4.4%	3.6%	2.0%	1.3%

Table 7: Vary dimensionality; average error over 20 runs; median output of 6 sketches; known α.

Experiments (unknown α)



`Baseline`
(greedy algo.)
$\Theta(n)$ space

`Sketch`
(our algo.)
$\tilde{O}(1/\epsilon^2)$ space

`CellCount`:
(streaming algo. for comparison)
$\tilde{O}(1/\epsilon^2)$ space

Dataset: I500k100x5d

Open problems

k-site model

- A number of bounds can possibly be improved. For example:
  - Can we get the optimal upper bound $\tilde{O}(k/\epsilon^2)$ for F_0?
  - Can we remove the k^3 factor in the communication cost for F_p?

Open problems

k-site model

- A number of bounds can possibly be improved. For example:
  - Can we get the optimal upper bound $\tilde{O}(k/\epsilon^2)$ for F_0?
  - Can we remove the k^3 factor in the communication cost for F_p?

- Can we obtain efficient algorithms for L_p-sampling?

Open problems

k-site model

- A number of bounds can possibly be improved. For example:
  - Can we get the optimal upper bound $\tilde{O}(k/\epsilon^2)$ for F_0?
  - Can we remove the k^3 factor in the communication cost for F_p?

- Can we obtain efficient algorithms for L_p-sampling?

- Lower bounds?

Open problems

k-site model

- A number of bounds can possibly be improved. For example:
  - Can we get the optimal upper bound $\tilde{O}(k/\epsilon^2)$ for F_0?
  - Can we remove the k^3 factor in the communication cost for F_p?

- Can we obtain efficient algorithms for L_p-sampling?

- Lower bounds?

Streaming model

- Algorithms for general metrics?
  (Now we can only do for some specific metrics use LSHs)

Thank you!
Questions?

– Communication-Efficient Computation on Distributed Noisy Datasets
Zhang, SPAA 2015

– Streaming Algorithms for Robust Distinct Elements
Chen and Zhang, SIGMOD 2016