Continuous Matrix Approximation On Distributed Data

Mina Ghashami
Jeff M. Phillips
Feifei Li

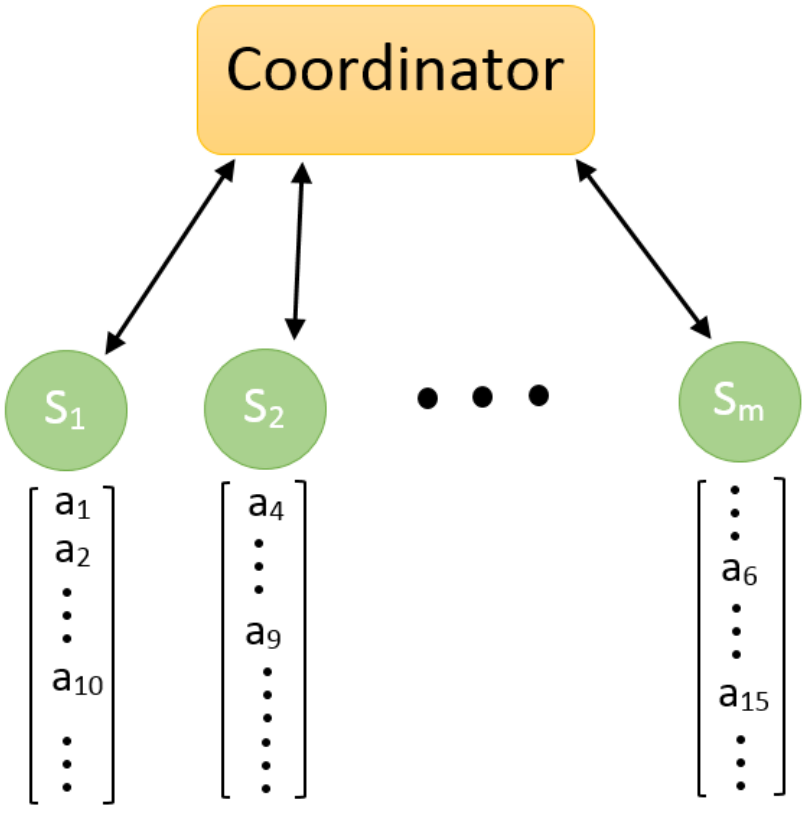
THE UNIVERSITY OF UTAH

Distributed Streaming Model

- m distributed sites $S_1, ..., S_m$
- Each observes a disjoint stream
- We denote the union of all streams as $A = (a_1, ..., a_n)$
- a single coordinator C

The Goal is to:

1. monitor a function f at C
2. minimizing space usage at sites and C
3. minimizing communication between sites and C



Two Related Problems

- **Distributed Streaming Matrix Approximation**: Here $A \in \mathbb{R}^{n \times d}$ and each a_i is a record of length d.The goal is to continuously track a small approximation to matrix A.

- **Heavy Hitters in Distributed Streams**: Here each a_i is an element of a bounded universe $[u] = \{1, ..., u\}$ where $f_e(A)$ is the frequency of e in A. The goal is to find ϕ-heavy hitters of A for some $\phi \in [0, 1]$; which are those elements with $f_e(A) \geq \phi n$.

  - ε-**approximate** ϕ-**heavy hitters**:
    Contains elements with $f_e(A) \geq \phi n$,
    Might contain elements with $(\phi - \varepsilon)n \leq f_e(A) \leq \phi n$,
    Does not contain elements with $f_e(A) \leq (\phi - \varepsilon)n$.

Matrix Approximation in Centralized Stream

Frequent Directions is the state of the art:

- It maintains a sketch $B \in \mathbb{R}^{\ell \times d}$ with only $\ell \ll n$ rows,
- Uses $O(d\ell)$ space and runs in $O(nd\ell)$
- Guarantees that $A^T A \approx B^T B$. More precisely, for $\forall |x| = 1$:

$$0 \leq \|Ax\|^2 - \|Bx\|^2 \leq 2\|A\|_F^2 / \ell$$

- Also gives a projection error bound:

$$\|A - \pi_{B_k}(A)\|_F^2 \leq (1+\varepsilon)\|A - A_k\|_F^2$$

  where $\pi_{B_k}(A)$ is the projection of A onto the row-space of B_k.
- Two FD sketches can be merged in $O(d\ell^2)$ time.

No work on matrix approximation in distributed streams.

Heavy Hitters in Distributed Stream

Three main works on this problem, all on unit-weight setting:

- Sampling by Cormode *et al.*: Each site samples some items from their stream and send it to C.
- Deterministic by Yi and Zhang: A threshold-based communication, each site keeps track of number of items it receives.
- Randomized by Huang *et al.*: For each item a in the stream a site chooses to send a message to C with a probability $p = \sqrt{m}/(\varepsilon\hat{n})$ where $\hat{n}$ is a 2-approximation of the total count.

No work on tracking heavy hitters in weighted distributed streams

Weighted Heavy Hitters in Distributed Streams

The model:

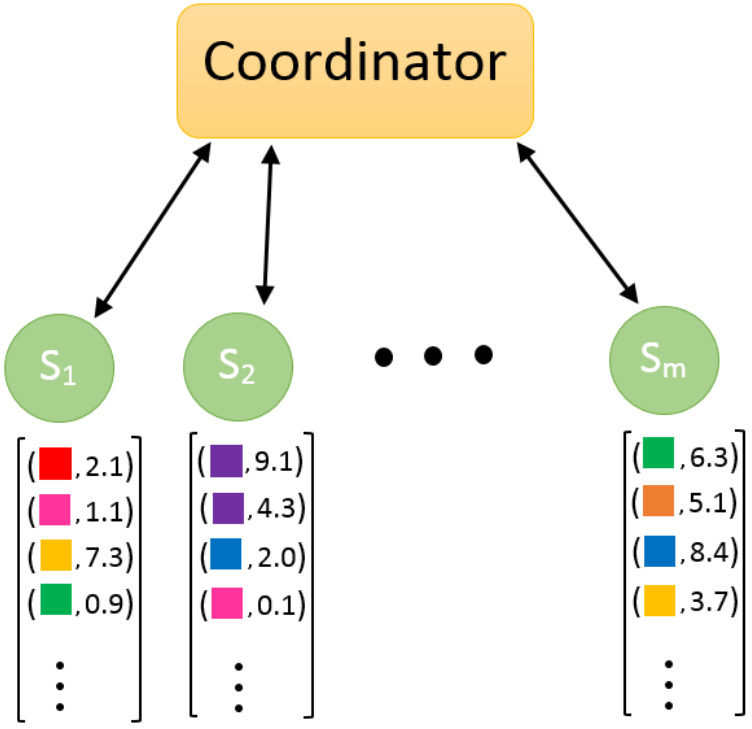
- $A = \{(a_1, w_1), (a_2, w_2), ...\}$
- Item $a \in [u]$, weight $w \in \mathbb{R}$

  [u] = {■, ■, ■, ■, ■, ■, ■}

- Define $f_e(A) = \sum_{(e,w_i)\in A} w_i$

  $f_{■}(A) = 9.1 + 4.3 = 13.4$

- Define $W_A = \sum_{(a_i,w_i)\in A} w_i$



Weighted Heavy Hitters in Distributed Streams

The model:

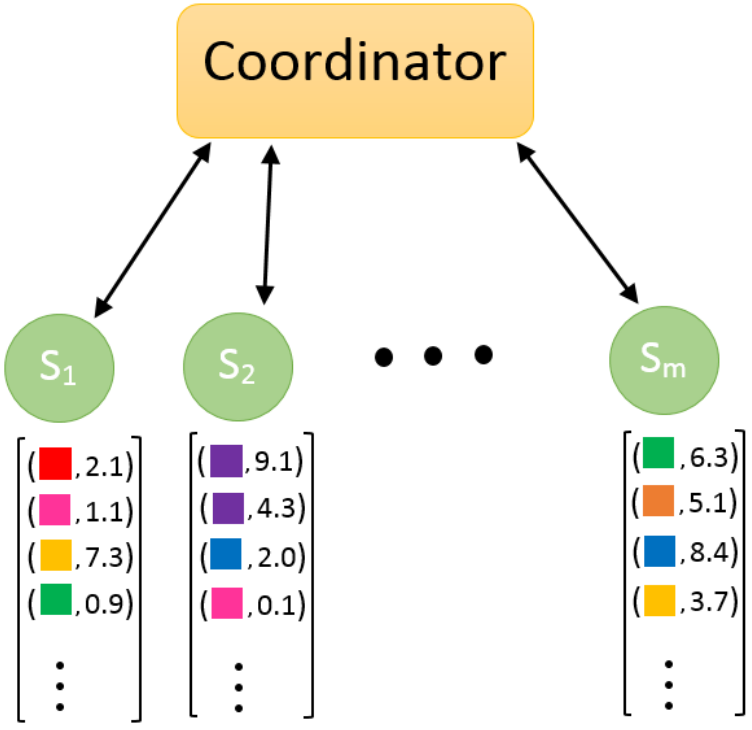
- $A = \{(a_1, w_1), (a_2, w_2), ...\}$
- Item $a \in [u]$, weight $w \in \mathbb{R}$

[u] = {■, ■, ■, ■, ■, ■, ■}

- Define $f_e(A) = \sum_{(e,w_i)\in A} w_i$

$f_{■}(A) = 9.1 + 4.3 = 13.4$

- Define $W_A = \sum_{(a_i,w_i)\in A} w_i$



Weighted Heavy Hitters problem:

- The goal is to approximate $\hat{f}_e$ at coordinator such that:

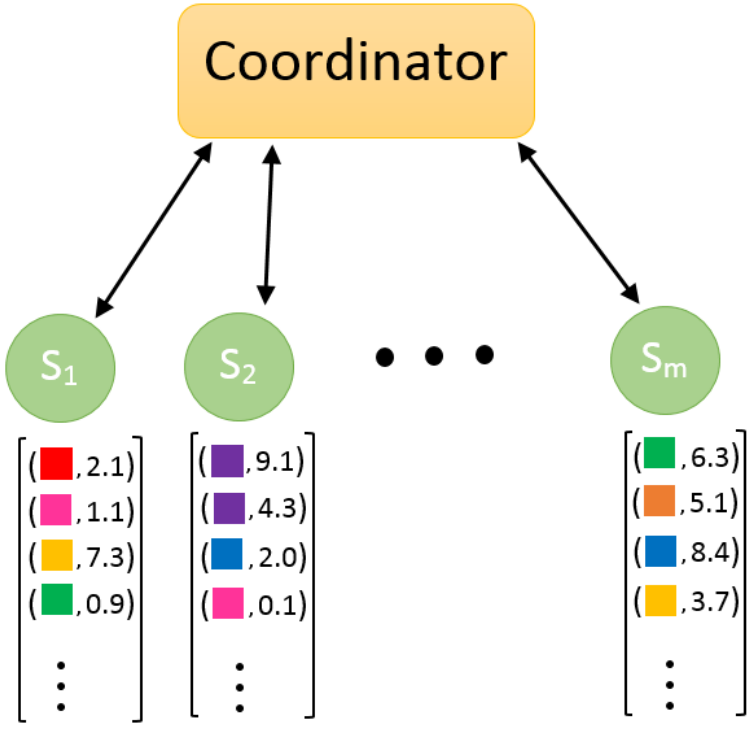
$$|\hat{f}_e(A) - f_e(A)| \leq \varepsilon W_A$$

- While minimizing the communication (#messages) with C

We have to upper bound weights

To avoid infinite communication...

- We put an upper bound β on weight of each item

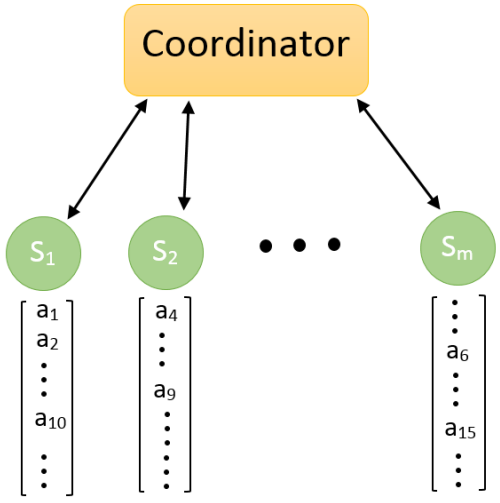


- If size of stream is N, then $W_A \leq \beta N$

Weighted Heavy Hitters: Protocol 1

An intuitive approach:

- Each site counts frequencies **locally**
- **Periodically** send the results to C
- C aggregates and **update** sites

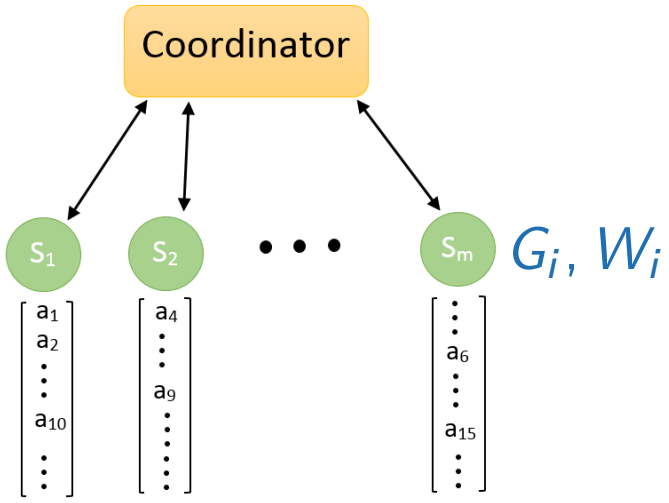


Weighted Heavy Hitters: Protocol 1

For simplicity: $\hat{W}$ = total weight of first few items is known by all sites.

P1 runs in rounds:

- Each site counts frequencies locally



Algorithm 1. P1 at site S_i

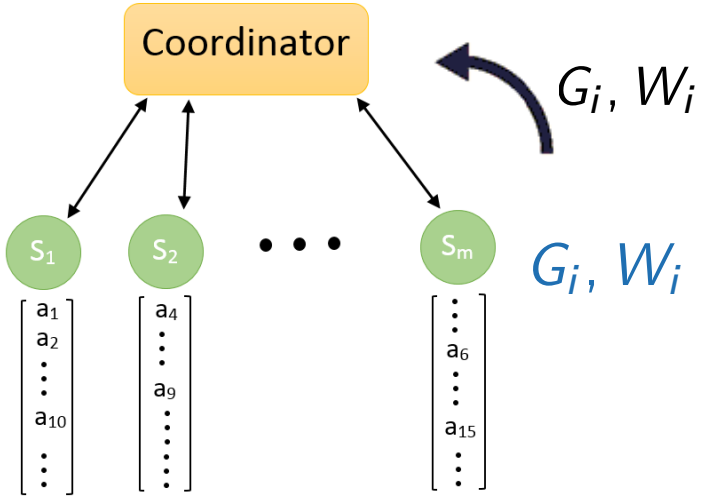
for (a_n, w_n) in round j **do**
  Update $G_i \leftarrow \mathrm{MG}_{\varepsilon'}(G_i, (a_n, w_n))$
  Update $W_i \mathrel{+}= w_n$
  if $(W_i \geq \tau = (\varepsilon/2m)\hat{W})$ **then**
    Send (G_i, W_i) to C
    Make G_i and W_i empty

Weighted Heavy Hitters: Protocol 1

For simplicity: $\hat{W}$ = total weight of first few items is known by all sites.

P1 runs in rounds:

- Each site counts frequencies locally
- Periodically send the results to C



Algorithm 1. P1 at site S_i

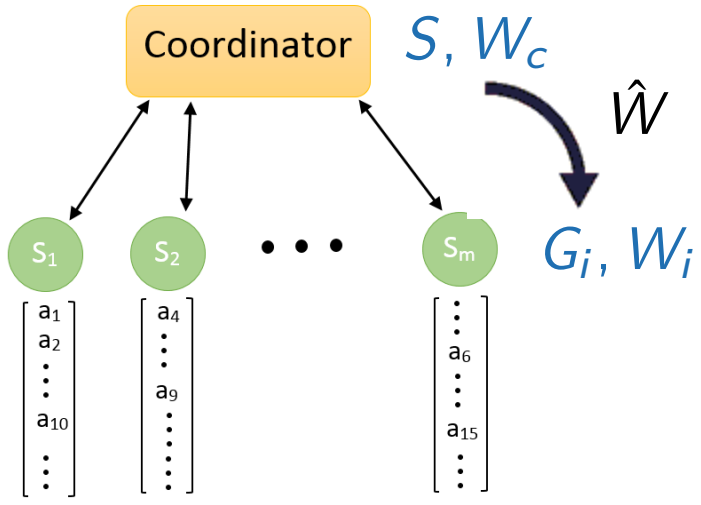
for (a_n, w_n) in round j **do**
  Update $G_i \leftarrow \mathrm{MG}_{\varepsilon'}(G_i, (a_n, w_n))$
  Update $W_i \mathrel{+}= w_n$
  if $(W_i \geq \tau = (\varepsilon/2m)\hat{W})$ **then**
    Send (G_i, W_i) to C
    Make G_i and W_i empty

Weighted Heavy Hitters: Protocol 1

For simplicity: $\hat{W}$ = total weight of first few items is known by all sites.

P1 runs in rounds:

- Each site counts frequencies locally
- Periodically send the results to C
- C aggregates and updates sites



Algorithm 1. P1 at site S_i

for (a_n, w_n) in round j **do**
  Update $G_i \leftarrow \mathrm{MG}_{\varepsilon'}(G_i, (a_n, w_n))$
  Update $W_i \mathrel{+}= w_n$
  if $(W_i \geq \tau = (\varepsilon/2m)\hat{W})$ **then**
    Send (G_i, W_i) to C
    Make G_i and W_i empty

Algorithm 2. P1 at site C

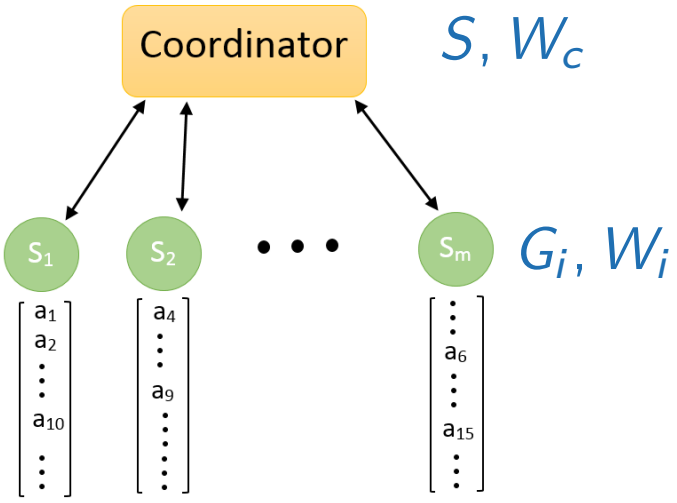
On input (G_i, W_i):
Update $S \leftarrow \mathrm{Merge}_{\varepsilon'}(S, G_i)$
Update $W_C \mathrel{+}= W_i$
if $(W_C/\hat{W} > 1 + \varepsilon/2)$ **then**
  Update $\hat{W} \leftarrow W_C$
  Broadcast $\hat{W}$ to all sites

Weighted Heavy Hitters: Protocol 1

For simplicity: $\hat{W}$ = total weight of first few items is known by all sites.

P1 runs in rounds:

- Each site counts frequencies locally
- Periodically send the results to C
- C aggregates and updates sites



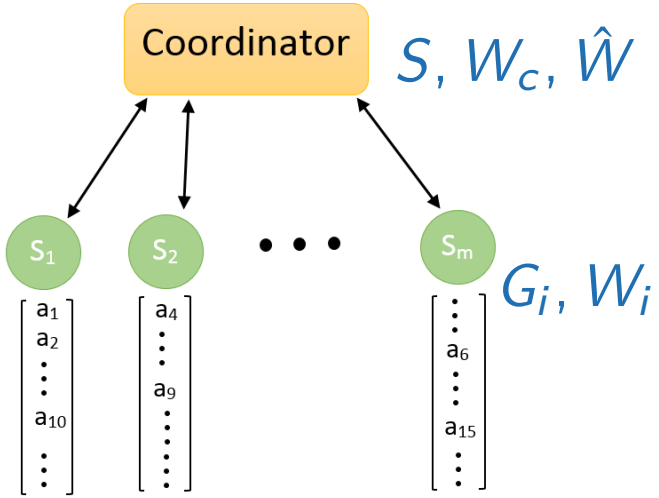
P1 maintain that for any item $e \in [u]$:

- $|f_e(S) - f_e(A)| \leq \varepsilon W_A$
- Communication cost is $O(\frac{m}{\varepsilon^2} \log(\beta N))$

Weighted Heavy Hitters: Protocol 2

We can reduce the communication to $O(\frac{m}{\varepsilon}\log(\beta N))$ by sending only **heavy counts** instead of the whole G_i.

[Yi,Zhang Algorithmca'13]



Algorithm 3. P2 at site S_i

for each item (a_n, w_n) **do**
  $W_i \mathrel{+}= w_n$ and $\Delta_{a_n} \mathrel{+}= w_n$.
  if $(W_i \geq (\varepsilon/m)\hat{W})$ **then**
    Send (total, W_i) to C
    Reset $W_i = 0$
  if $(\Delta_{a_n} \geq (\varepsilon/m)\hat{W})$ **then**
    Send (a_n, Δ_{a_n}) to C
    Reset $\Delta_{a_n} = 0$

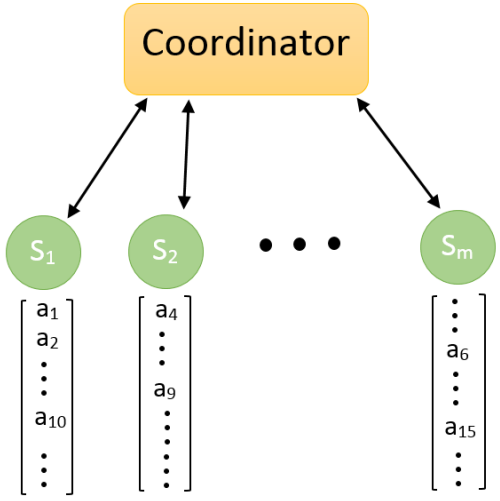
Algorithm 4. P2 at site S_i

On message (total, W_i):
Set $\hat{W} \mathrel{+}= W_i$
Set $\#\text{msg} \mathrel{+}= 1$
if $(\#\text{msg} \geq m)$ **then**
  Set $\#\text{msg} = 0$
  Broadcast $\hat{W}$ to all sites
On message (a_n, Δ_n):
Set $\hat{W}_{a_n} \mathrel{+}= \Delta_{a_n}$.

Weighted Heavy Hitters: Protocol 3

Sampling based protocol:($s = O(\frac{1}{\varepsilon^2})$)

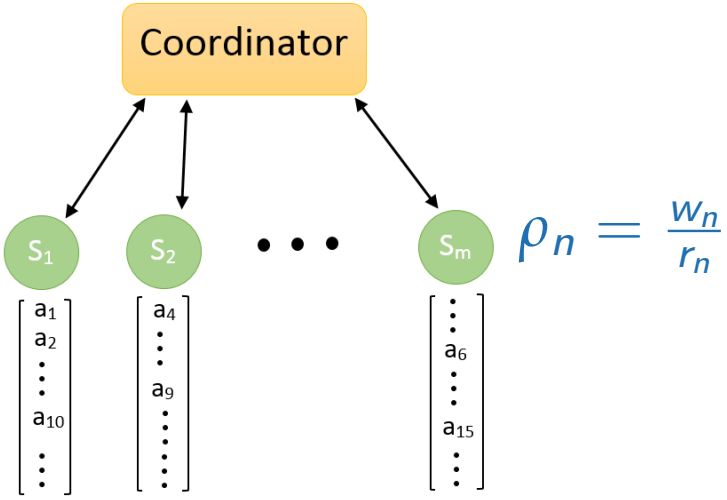
- It happens in rounds
- C fixes a threshold τ (initially $\tau = 1$)
  At round j, $\tau_j = 2^j$



Weighted Heavy Hitters: Protocol 3

Sampling based protocol:($s = O(\frac{1}{\varepsilon^2})$)

- It happens in rounds
- C fixes a threshold τ (initially $\tau = 1$)
  At round j, $\tau_j = 2^j$



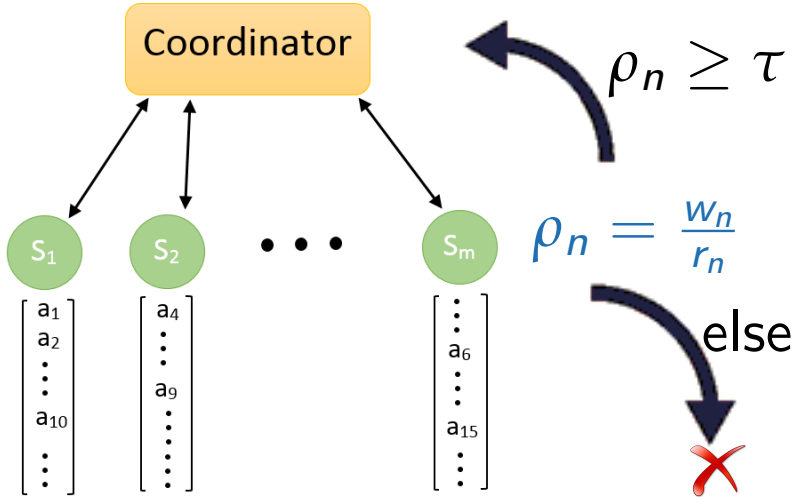
Algorithm 5. P3 at site S_i

for (a_n, w_n) in round j **do**
choose $r_n \in \text{Unif}(0,1)$ and set $\rho_n = w_n / r_n$.
if $\rho_n \geq \tau$ **then** send (a_n, w_n, ρ_n) to C

Weighted Heavy Hitters: Protocol 3

Sampling based protocol:($s = O(\frac{1}{\varepsilon^2})$)

- It happens in rounds
- C fixes a threshold τ (initially $\tau = 1$)
  At round j, $\tau_j = 2^j$



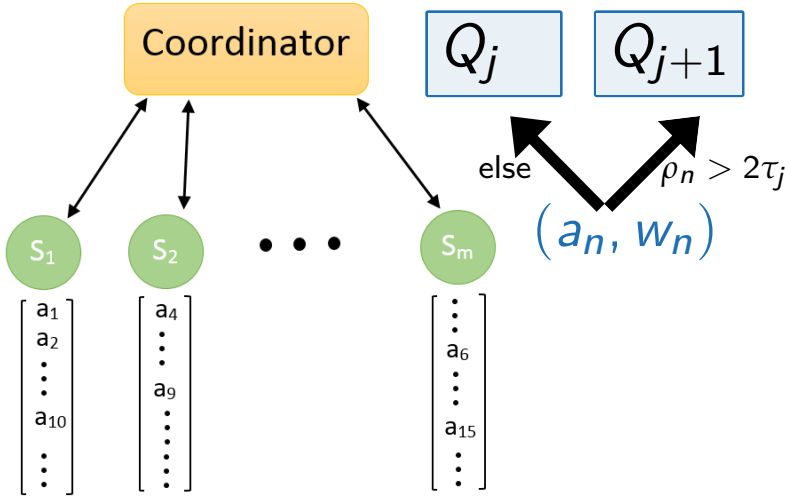
Algorithm 5. P3 at site S_i

for (a_n, w_n) in round j **do**
choose $r_n \in \text{Unif}(0, 1)$ and set $\rho_n = w_n / r_n$.
if $\rho_n \geq \tau$ **then** send (a_n, w_n, ρ_n) to C

Weighted Heavy Hitters: Protocol 3

Sampling based protocol:($s = O(\frac{1}{\varepsilon^2})$)

- It happens in rounds
- C fixes a threshold τ (initially $\tau = 1$)
  At round j, $\tau_j = 2^j$



Algorithm 5. P3 at site S_i

for (a_n, w_n) in round j **do**
  choose $r_n \in \text{Unif}(0,1)$ and set $\rho_n = w_n / r_n$.
  if $\rho_n \geq \tau$ **then** send (a_n, w_n, ρ_n) to C

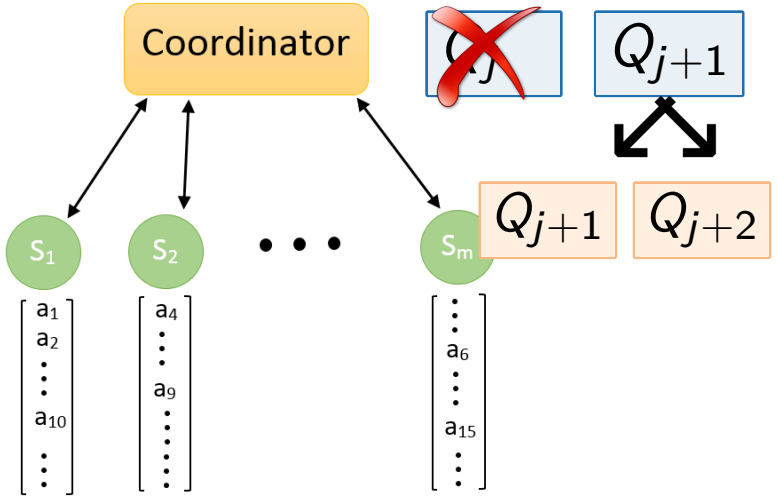
Algorithm 6. P3 at site C

if $\rho > 2\tau_j$ **then** put a_n in Q_{j+1}, **else** put a_n in Q_j.
if $|Q_{j+1}| \geq s$ **then**
  Set $\tau_{j+1} = 2\tau_j$; broadcast τ_{j+1}.
  for $(a_n, w_n, \rho_n) \in Q_{j+1}$ **do**
    if $\rho_n > 2\tau_{j+1}$, put a_n in Q_{j+2}.

Weighted Heavy Hitters: Protocol 3

Sampling based protocol:($s = O(\frac{1}{\varepsilon^2})$)

- It happens in rounds
- C fixes a threshold τ (initially $\tau = 1$)
  At round j, $\tau_j = 2^j$



Algorithm 5. P3 at site S_i

for (a_n, w_n) in round j **do**
  choose $r_n \in \text{Unif}(0,1)$ and set $\rho_n = w_n / r_n$.
  if $\rho_n \geq \tau$ **then** send (a_n, w_n, ρ_n) to C

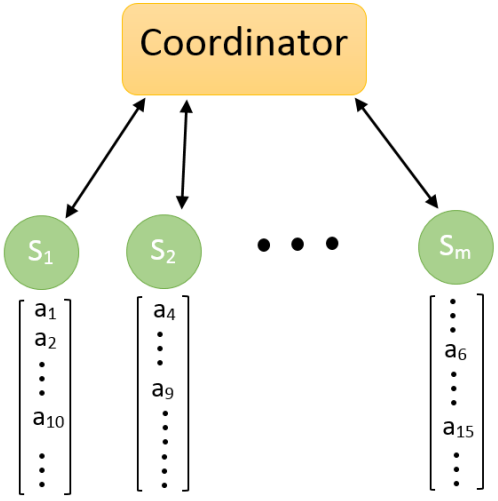
Algorithm 6. P3 at site C

if $\rho > 2\tau_j$ **then** put a_n in Q_{j+1}, **else** put a_n in Q_j.
if $|Q_{j+1}| \geq s$ **then**
  Set $\tau_{j+1} = 2\tau_j$; broadcast τ_{j+1}.
  for $(a_n, w_n, \rho_n) \in Q_{j+1}$ **do**
    if $\rho_n > 2\tau_{j+1}$, put a_n in Q_{j+2}.

Weighted Heavy Hitters: Protocol 3

Sampling based protocol:($s = O(\frac{1}{\varepsilon^2})$)

- It happens in rounds
- C fixes a threshold τ (initially $\tau = 1$)
  At round j, $\tau_j = 2^j$



Maintain a sample of size $s = \frac{1}{\varepsilon^2} \log \frac{1}{\delta}$:

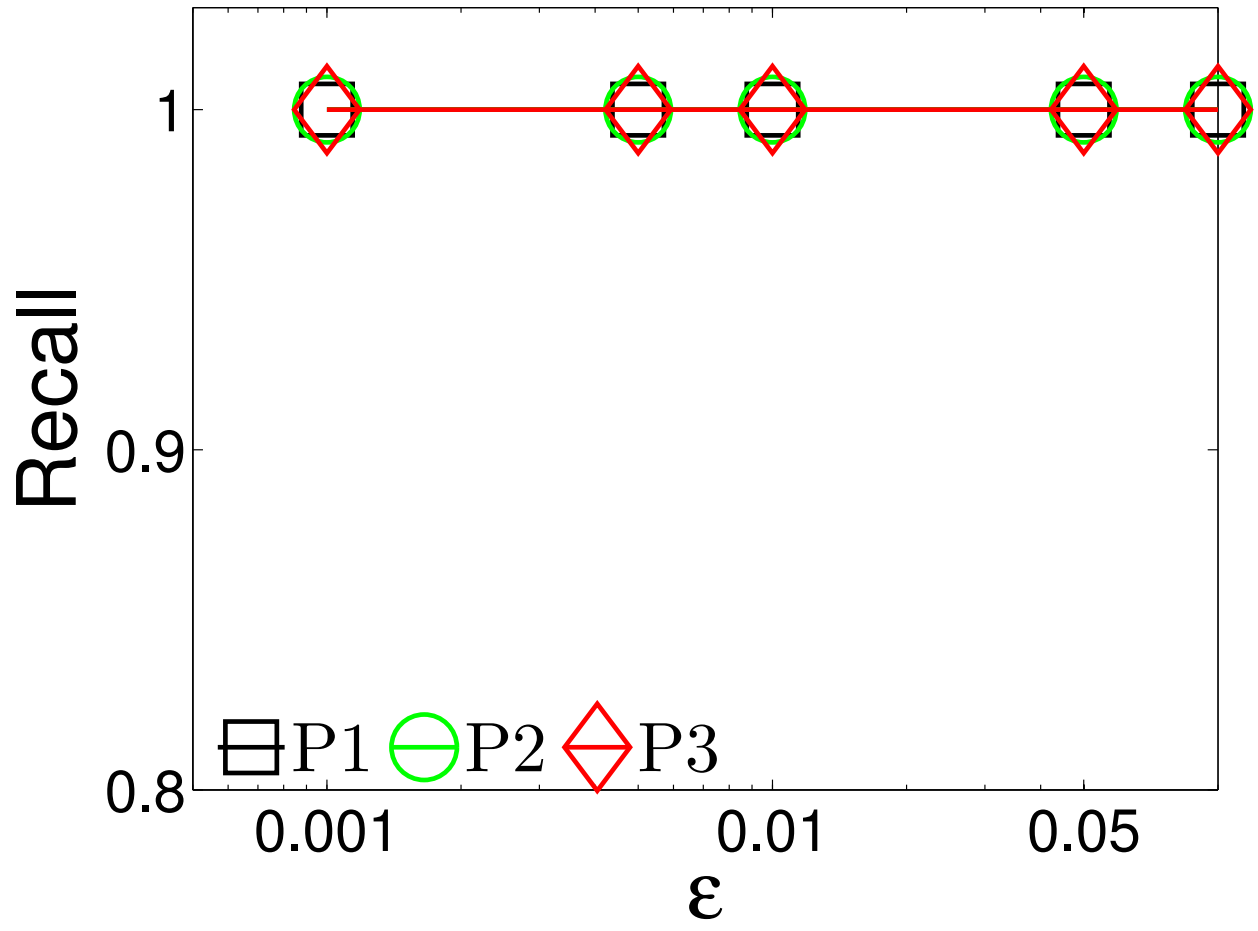
- For any item $e \in [u]$: $|f_e(S) - f_e(A)| \leq \varepsilon W_A$
- Communication cost would be $O((m+s)\log(\beta N))$

Experiments on Heavy Hitters

- We generated data from **Zipfian** distribution, and set the skew parameter to 2
- The dataset contained 10^7 points, in order to assign them weights we fixed the upper bound (default $\beta = 1000$)
- All of our heavy hitters protocols return an element e as heavy hitter only if $\hat{f}_e / \hat{W} \geq \phi - \varepsilon/2$, where $\phi = 0.05$

Experiments on Heavy Hitters

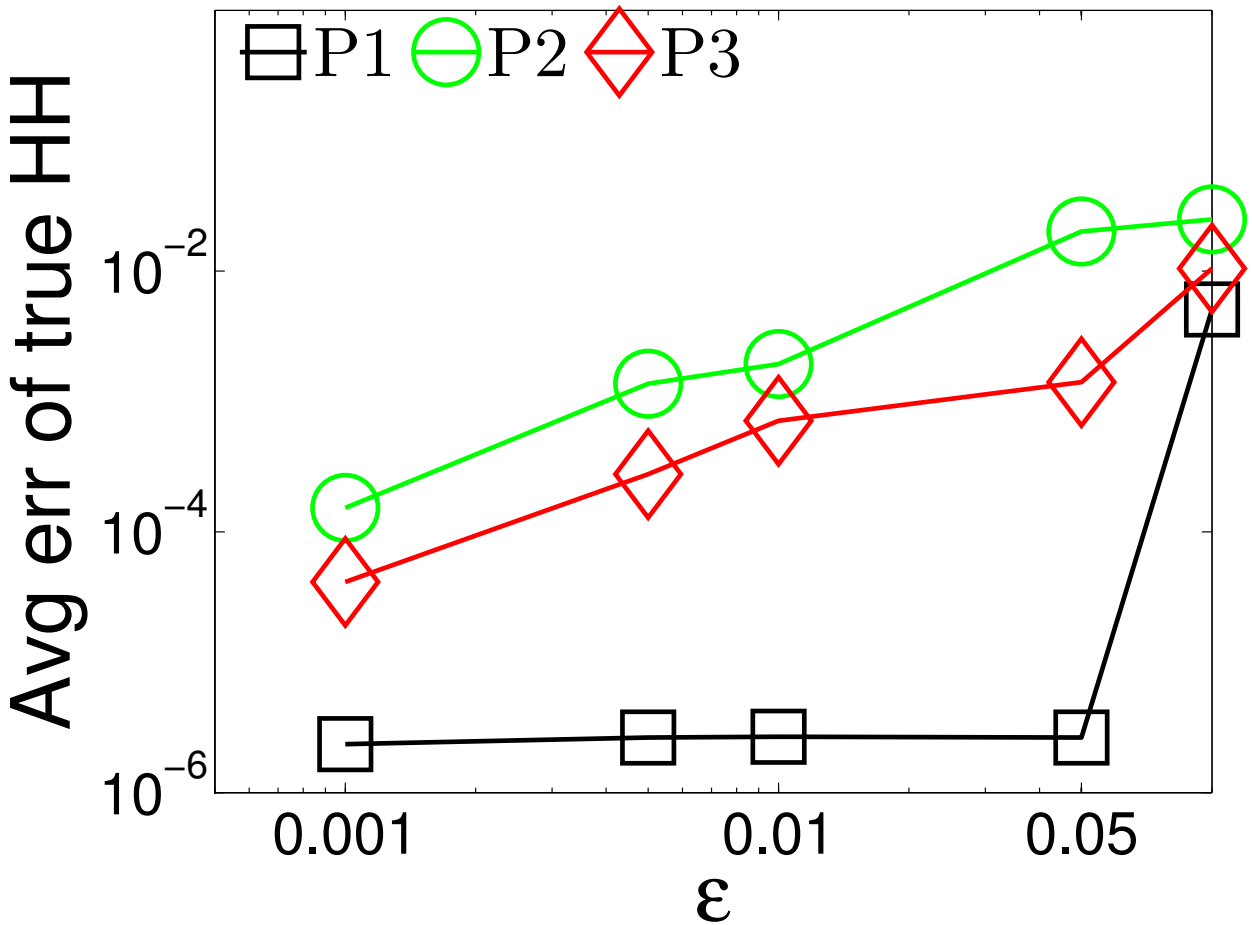
$$\text{Recall} = \frac{\text{The number of true heavy hitters returned by a protocol}}{\text{the correct number of true heavy hitters}}$$



Experiments on Heavy Hitters

plotted err $= \text{Avg}(|\hat{f}_e/W - f_e/W|)$

our error bound $= |\hat{f}_e - f_e| \leq \varepsilon W$



Distributed Matrix Tracking

- Each element a_n of a stream is now a row of the matrix
- Weight of each element is $w_n = \|a_n\|^2$ and bounded by β
- Coordinator's job is to maintain a much smaller matrix $B \in \mathbb{R}^{\ell \times d}$ as an approximation to A such that $\forall x \in \mathbb{R}^{d \times 1}$ (with $\|x\| = 1$):

$$|\|Ax\|^2 - \|Bx\|^2| \leq \varepsilon \|A\|_F^2.$$

- The above expression is equivalent to

$$\|A^T A - B^T B\|_2 \leq \varepsilon \|A\|_F^2.$$

Thus, the approximation guarantee shows that the covariance of A is well-approximated by B.

Distributed Matrix Tracking Protocol 1

Extension of $P1$ in heavy hitter tracking...

Algorithm 0.1 P1: Deterministic Matrix Tracking (at S_i)

for (a_n, w_n) in round j **do**
  Update $B_i \leftarrow \text{FD}_{\varepsilon'}(B_i, a_n)$; and $F_i \mathrel{+}= \|a_n\|^2$.
  if $(F_i \geq \tau = (\varepsilon/2m)\hat{F})$ **then**
    Send (B_i, F_i) to coordinator; make B_i, F_i empty.

Algorithm 0.2 P1: Deterministic Matrix Tracking (at C)

On input (B_i, F_i):
Update sketch $B \leftarrow \text{Merge}_{\varepsilon'}(B, B_i)$ and $F_C \mathrel{+}= F_i$.
if $(F_C/\hat{F} > 1 + \varepsilon/2)$ **then**
  Update $\hat{F} \leftarrow F_C$, and broadcast $\hat{F}$ to all sites.

Communication cost is $O(\frac{m}{\varepsilon^2} \log \beta N)$

Distributed Matrix Tracking Protocol 2

Extension of $P2$ in heavy hitter tracking...with $O(\frac{m}{\varepsilon}\log \beta N)$ msgs

Algorithm 0.3 P2: Deterministic Matrix Tracking (at S_j)

$F_j \mathrel{+}= \|a_i\|^2$
if $(F_j \geq \frac{\varepsilon}{m}\hat{F})$ **then** Send F_j to coordinator; set $F_j = 0$.
Set $B_j \leftarrow [B_j; a_i]$, $[U, \Sigma, V] = \text{svd}(B_j)$
for $((v_\ell, \sigma_\ell)$ such that $\sigma_\ell^2 \geq \frac{\varepsilon}{m}\hat{F})$ **do**
  Send $\sigma_\ell v_\ell$ to coordinator; set $\sigma_\ell = 0$.
$B_j = U\Sigma V^T$

Algorithm 0.4 P2: Deterministic Matrix Tracking (at C)

On a scalar message F_j from site S_j:
Set $\hat{F} \mathrel{+}= F_j$ and $\#\text{msg} \mathrel{+}= 1$.
if $(\#\text{msg} \geq m)$ **then**
  Set $\#\text{msg} = 0$ and broadcast $\hat{F}$ to all sites.
On a vector message $r = \sigma v$: append $B \leftarrow [B; r]$

Distributed Matrix Tracking Protocol 3

Extension of $P3$ from HH tracking, with $O((m + \frac{1}{\varepsilon^2}) \log \beta N)$ msgs

Algorithm 0.5 P3: Matrix Tracking (at S_i)

for a_n in round j **do**
  choose $r_n \in \text{Unif}(0, 1)$ and set $\rho_n = \|a_n\|^2 / r_n$.
  if $\rho_n \geq \tau$ **then** send $(a_n, \|a_n\|^2, \rho_n)$ to C.

Algorithm 0.6 P3: Matrix Tracking (at C)

On input of $(a_n, \|a_n\|^2, \rho_n)$ from any site in round j:
if $\rho > 2\tau_j$ **then** put a_n in Q_{j+1}, **else** put a_n in Q_j.
if $|Q_{j+1}| \geq s$ **then**
  Set $\tau_{j+1} = 2\tau_j$; broadcast τ_{j+1} to all sites.
  for $(a_n, \|a_n\|^2, \rho_n) \in Q_{j+1}$ **do**
    if $\rho_n > 2\tau_{j+1}$, put a_n in Q_{j+2}.

Experiments on Distributed Matrix Tracking

Two large datasets...

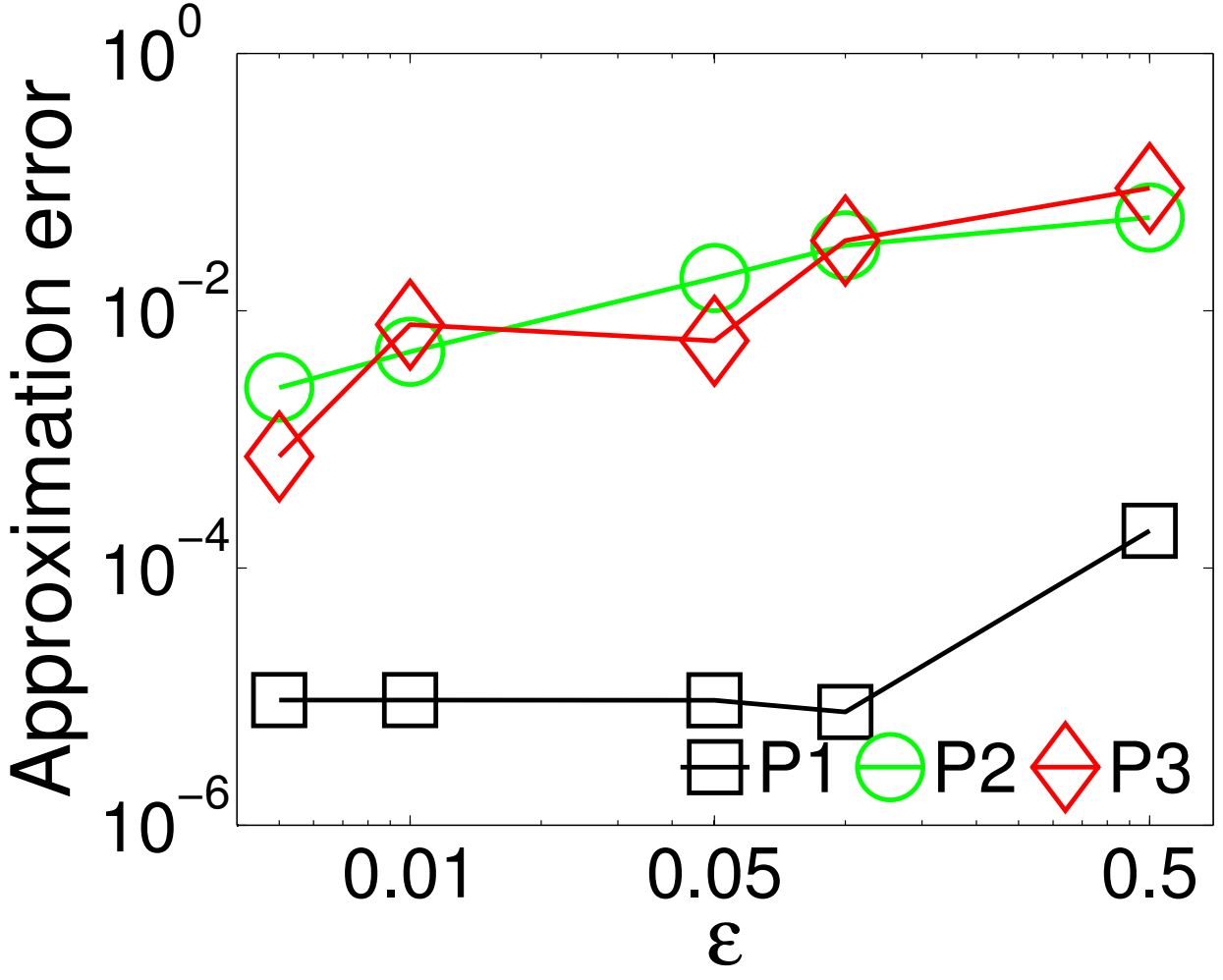
Dataset	Rows	Columns	Description
PAMAP	629250	44	Physical Activity Monitoring
YearPredictionMSD	500000	90	Million Songs

Each formed an $N \times d$ matrix, distributed among m nodes

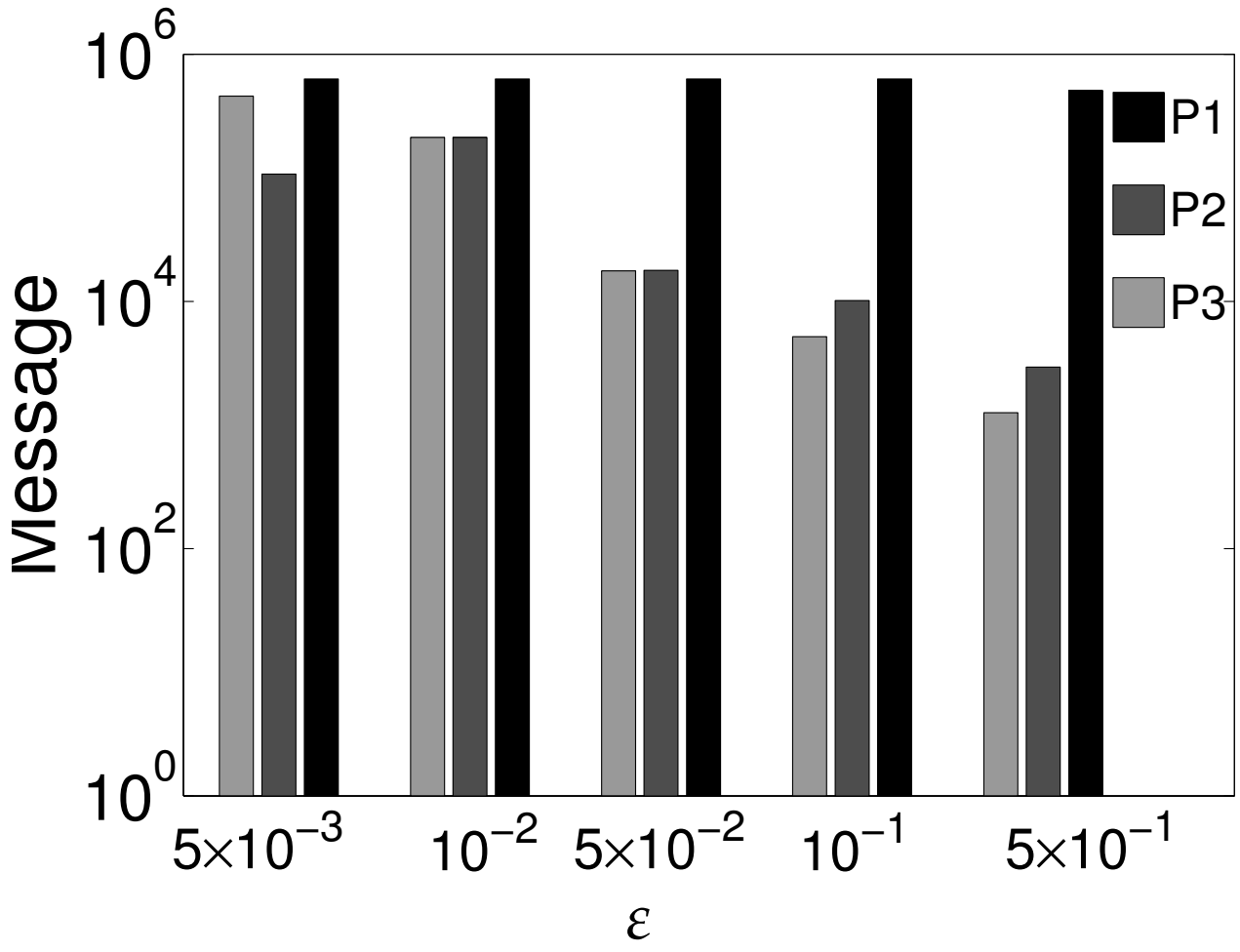
Experiments on Distributed Matrix Tracking

Approximation Error $= \max_{x,\|x\|=1}(\|Ax\|^2 - \|Bx\|^2)/\|A\|_F^2$

Theoretical Bound $= (\|Ax\|^2 - \|Bx\|^2)/\|A\|_F^2 \leq \varepsilon$



Experiments on Distributed Matrix Tracking



Summary

- Three approaches for distributed weighted heavy hitter tracking and low rank matrix approximation

Protocol	Communication
P1	$O(\frac{m}{\varepsilon^2} \log \beta N)$
P2	$O(\frac{m}{\varepsilon} \log \beta N)$
P3	$O((m + \frac{1}{\varepsilon^2}) \log \beta N)$

- All approaches are simple, easy to implement, work pretty well, but tricky to analyze