Fractional Set Cover in the Streaming Model

Piotr Indyk
MIT

Sepideh Mahabadi
Columbia University

Ronitt Rubinfeld
MIT/TAU

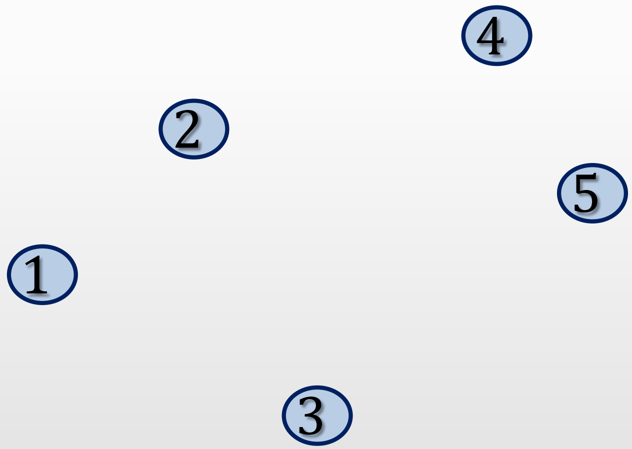
Jonathan Ullman
Northeastern Univ

Ali Vakilian
MIT

Anak Yodpinyanee
MIT

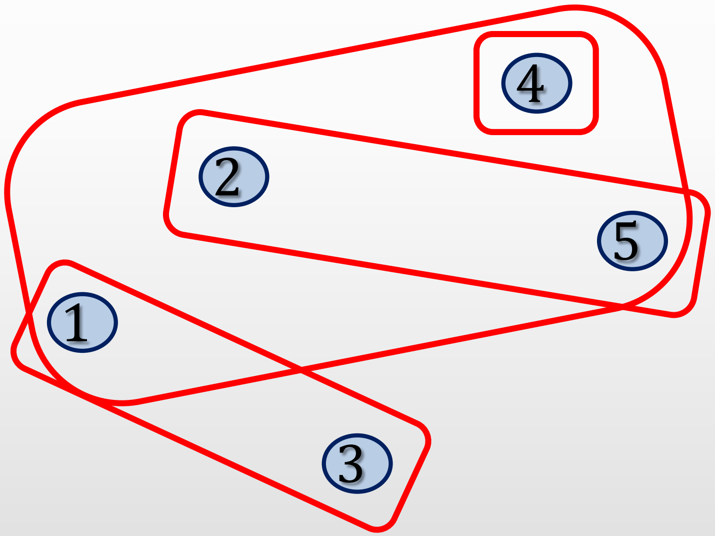
Set Cover Problem

- Input: Collection $\mathcal{F}$ of sets $S_1, \dots, S_m$, each a subset of $\mathcal{U} = \{1, \dots, n\}$



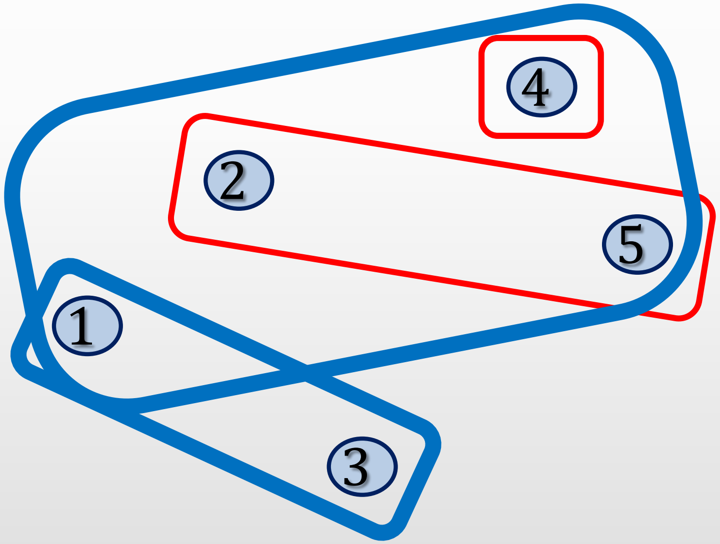
Set Cover Problem

- Input: Collection $\mathcal{F}$ of sets $S_1, \ldots, S_m$, each a subset of $\mathcal{U} = \{1, \ldots, n\}$



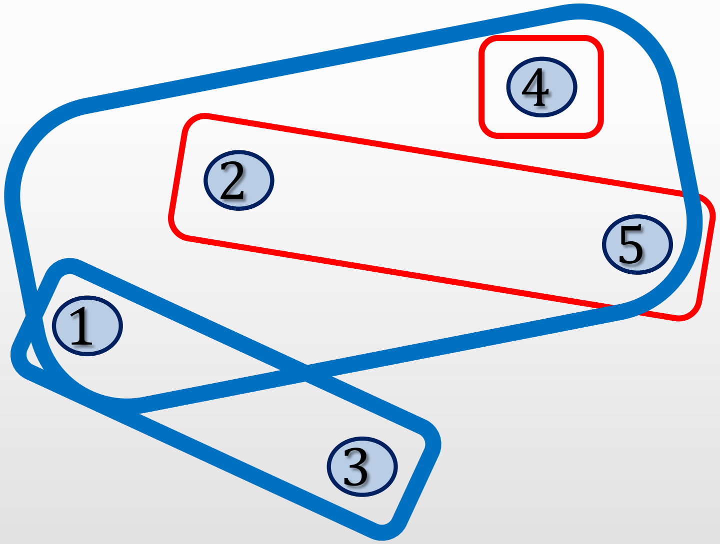
Set Cover Problem

- Input: Collection $\mathcal{F}$ of sets $S_1, \dots, S_m$, each a subset of $\mathcal{U} = \{1, \dots, n\}$
- Output: a subset $\mathcal{C}$ of $\mathcal{F}$ such that:
  - o $\mathcal{C}$ covers $\mathcal{U}$
  - o $|\mathcal{C}|$ is minimized



Set Cover Problem

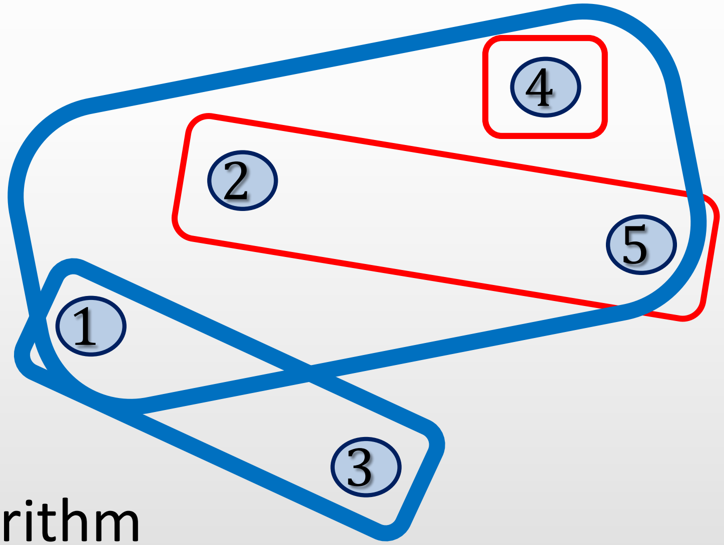
- Input: Collection $\mathcal{F}$ of sets $S_1, \dots, S_m$, each a subset of $\mathcal{U} = \{1, \dots, n\}$
- Output: a subset $\mathcal{C}$ of $\mathcal{F}$ such that:
  - $\mathcal{C}$ covers $\mathcal{U}$
  - $|\mathcal{C}|$ is minimized



- Complexity:
  - NP-hard

Set Cover Problem

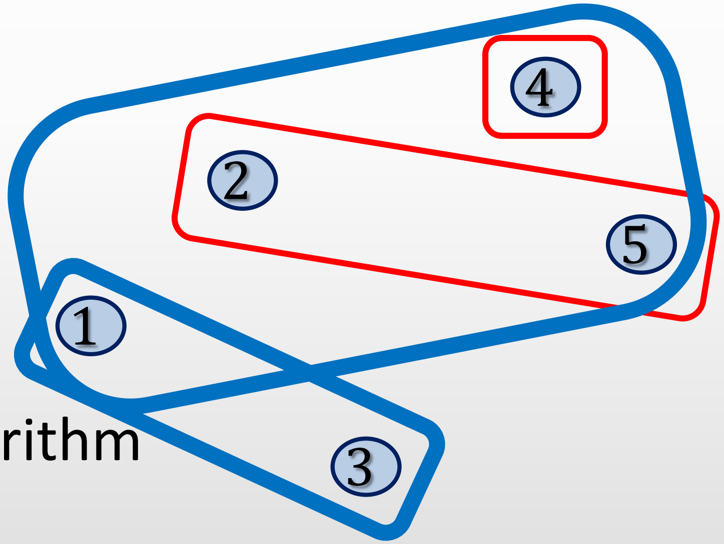
- Input: Collection $\mathcal{F}$ of sets $S_1, \ldots, S_m$, each a subset of $\mathcal{U} = \{1, \ldots, n\}$
- Output: a subset $\mathcal{C}$ of $\mathcal{F}$ such that:
    - $\mathcal{C}$ covers $\mathcal{U}$
    - $|\mathcal{C}|$ is minimized



- Complexity:
    - NP-hard
    - Greedy $(\ln n)$-approximation algorithm

Set Cover Problem

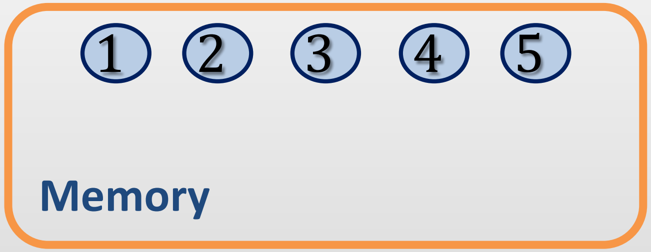
- Input: Collection $\mathcal{F}$ of sets $S_1, \dots, S_m$, each a subset of $\mathcal{U} = \{1, \dots, n\}$
- Output: a subset $\mathcal{C}$ of $\mathcal{F}$ such that:
  - $\mathcal{C}$ covers $\mathcal{U}$
  - $|\mathcal{C}|$ is minimized



- Complexity:
  - NP-hard
  - Greedy $(\ln n)$-approximation algorithm
  - Can't do better unless P=NP [LY91][RS97][Fei98][AMS06][DS14]

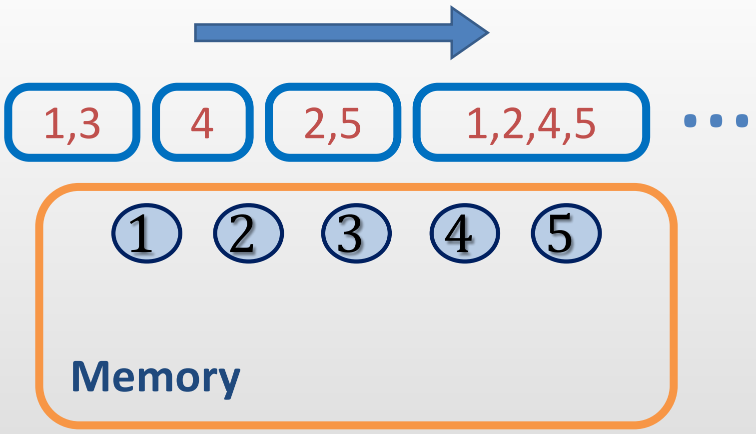
Streaming Set Cover

- Model [SG09]
  - Elements are store in the main memory



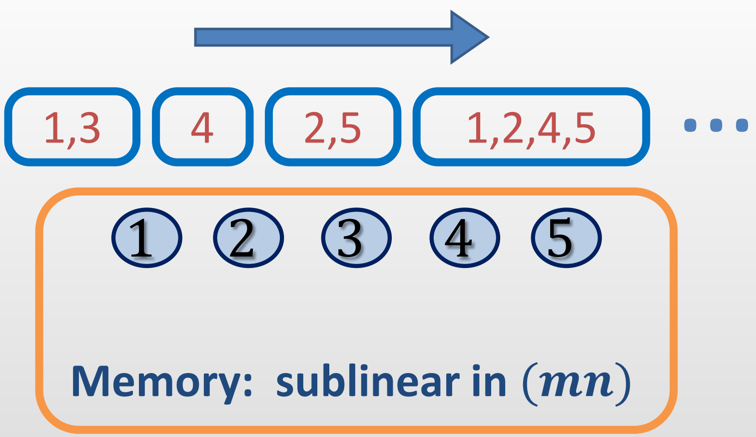
Streaming Set Cover

- Model [SG09]
  - Elements are store in the main memory
  - Sequential access to $S_1, S_2, \ldots, S_m$



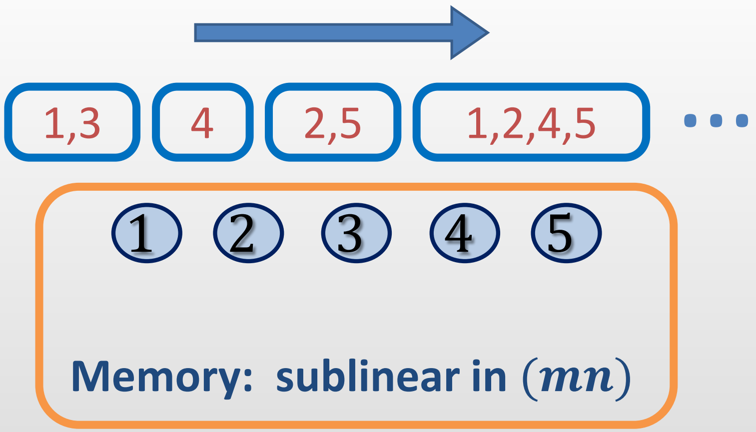
Streaming Set Cover

- Model [SG09]
  - Elements are store in the main memory
  - Sequential access to $S_1, S_2, \ldots, S_m$
  1. One (or few) passes
  2. Sublinear (i.e., $o(mn)$) storage
  3. (Hopefully) decent approximation factor



Streaming Set Cover

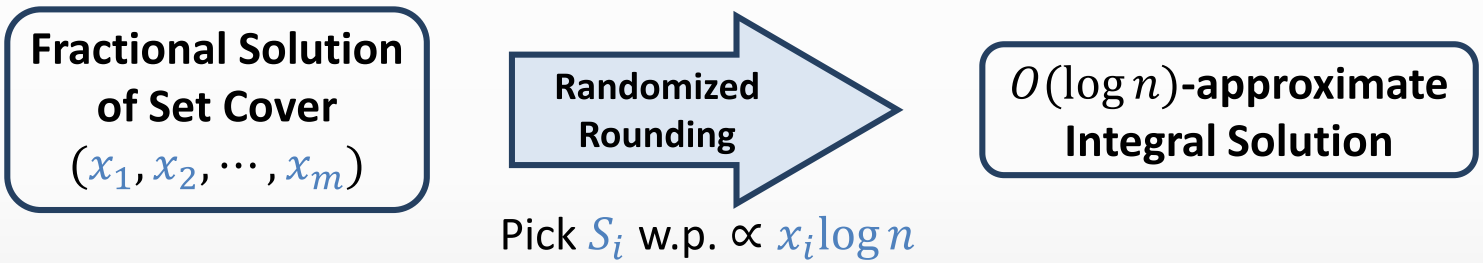
- Model [SG09]
  - o Elements are store in the main memory
  - o Sequential access to $S_1, S_2, \ldots, S_m$
  1. One (or few) passes
  2. Sublinear (i.e., $o(mn)$) storage
  3. (Hopefully) decent approximation factor



- Why?
  - o A classic optimization problem
  - o Application in "Big Data": Clustering, Topic Coverage

Fractional Set Cover

- Each set can be picked fractionally (assigning value $x_i \in [0,1]$ to each set S_i)



- The first step in solving covering LPs in stream
  - Packing LP (Fractional Maximum Matching)[AG11]

Previous and Our Results

INTEGRAL SET COVER	Approximation	Passes	Space
Greedy Algorithm	$O(\log n)$ $O(\log n)$	1 n	$O(mn)$ $O(n)$
[SG09]	$O(\log n)$	$O(\log n)$	$\tilde{O}(n)$
[ER14, CW16]	$O(n^\delta/\delta)$ $\Omega(n^\delta/\delta^2)$	$1/\delta - 1$	$\tilde{O}(n)$
[DIMV14, HIMV16, BEM17]	$O(\rho/\delta)$	$O(1/\delta)$	$\tilde{O}(mn^\delta)$
[AKL16, A17]	$1/\delta$	polylog	$\tilde{\Omega}(mn^\delta)$
FRACTIONAL SET COVER	$1+\varepsilon$	$O(1/\delta)$	$\tilde{O}(mn^{O(\delta/\varepsilon)})$

$\delta < 1$ (applies to rows [ER14, CW16] through [AKL16, A17])

ρ = approximation factor for offline **Set Cover**

$\tilde{O}(f(m,n)) = O(f(m,n)\,\varepsilon^{-c}\log^c m \log^c n)$

n = number of *elements*

m = number of *sets*

This Talk

Theorem: there exists a $(1+\epsilon)$ approximation algorithm for the fractional set cover problem in the streaming setting, with d passes, that uses $\tilde{O}(mn^{O(\frac{1}{d\epsilon})} + n)$ space.

The Plan

- The Multiplicative Weight Update framework
  - MWU for the Set Cover
  - The average constraint: Oracle
- Implement MWU Oracle Naively in the streaming
  - $O(\frac{k \log n}{\epsilon^2})$ passes
- Reducing the number of passes to logarithmic
  - Reducing Width via Extended Set System
  - Fractional Max *k*-Cover
- Reducing the number of passes to a constant
  - Running several rounds of MWU together by sampling in advance

The Plan

- **The Multiplicative Weight Update framework**
  - MWU for the Set Cover
  - The average constraint: Oracle
- Implement MWU Oracle Naively in the streaming
  - $O(\frac{k \log n}{\epsilon^2})$ passes
- Reducing the number of passes to logarithmic
  - Reducing Width via Extended Set System
  - Fractional Max *k*-Cover
- Reducing the number of passes to a constant
  - Running several rounds of MWU together by sampling in advance

MWU to solve LP

Algorithm:

- Instead of solving for all the constraints, solve for a weighted average constraint.
- Take the solution
- The less a constraint is satisfied, the less weight it gets for the next iteration
- Repeat the above for T iterations
- Report the average solution found over all iterations.
- $T = O(\phi \log n / \epsilon^2)$

CoveringLP($A_{n \times m}, c_m, b_n$)

Min $c^T x$

$$Ax \geq b$$
$$x \geq 0$$

Oracle($A_{n \times m}, c_m, b_n, p^t$)

Min $c^T x$

$$(w^t)^T Ax \geq (w^t)^T b$$
$$x \geq 0$$

$w^1 \leftarrow (1, \cdots, 1)$ ▷ uniform weights
For $t = 1, t \leq T$ **do** ▷ T iterations
  $x^t \leftarrow$ solution of Oracle ▷ avg constraint w.r.t. w^t
  $w^{t+1} \leftarrow \mathbf{Update}(w^t, x^t)$
  ▷ decrease weight of constraints oversatisfied by x^t
$\bar{x} = \mathbf{avg}(x_1, \cdots x_T)$

MWU Update Rule:
$w_e^{t+1} := w_e^t \left(1 - \boldsymbol{\varepsilon}/\boldsymbol{\phi}(A_e x^t - b_e)\right)$

$\forall i, t: -\phi \leq A_e x^t - b_e \leq \phi$

$\forall i: \quad A_e \bar{x} \geq b_e - \varepsilon$

MWU to solve LP

CoveringLP($A_{n \times m}, c_m, b_n$)

Min $c^T x$

$$Ax \geq b$$
$$x \geq 0$$

MWU to solve LP

Algorithm:

- Instead of solving for all the constraints, solve for a weighted average constraint.
- Take the solution
- The less a constraint is satisfied, the less weight it gets for the next iteration
- Repeat the above for T iterations
- Report the average solution found over all iterations.

CoveringLP($A_{n\times m}, c_m, b_n$)

Min $c^T x$

$$Ax \geq b$$
$$x \geq 0$$

Oracle($A_{n\times m}, c_m, b_n, p^t$)

Min $c^T x$

$$(w^t)^T Ax \geq (w^t)^T b$$
$$x \geq 0$$

MWU to solve LP

Algorithm:

- Instead of solving for all the constraints, solve for a weighted average constraint.
- Take the solution
- The less a constraint is satisfied, the less weight it gets for the next iteration
- Repeat the above for T iterations
- Report the average solution found over all iterations.

CoveringLP($A_{n\times m}, c_m, b_n$)

Min $c^T x$

$$Ax \geq b$$
$$x \geq 0$$

Oracle($A_{n\times m}, c_m, b_n, p^t$)

Min $c^T x$

$$(w^t)^T Ax \geq (w^t)^T b$$
$$x \geq 0$$

$w^1 \leftarrow (1, \cdots, 1)$ ⊳ uniform weights
For $t = 1, t \leq T$ **do** ⊳ T iterations
    $x^t \leftarrow$ solution of Oracle ⊳ avg constraint w.r.t. w^t

MWU to solve LP

Algorithm:

- Instead of solving for all the constraints, solve for a weighted average constraint.
- Take the solution
- The less a constraint is satisfied, the less weight it gets for the next iteration
- Repeat the above for T iterations
- Report the average solution found over all iterations.

CoveringLP($A_{n\times m}, c_m, b_n$)

Min $c^T x$

$$Ax \geq b$$
$$x \geq 0$$

Oracle($A_{n\times m}, c_m, b_n, p^t$)

Min $c^T x$

$$(w^t)^T Ax \geq (w^t)^T b$$
$$x \geq 0$$

$w^1 \leftarrow (1, \cdots, 1)$ ▷ uniform weights
For $t = 1, t \leq T$ **do** ▷ T iterations
  $x^t \leftarrow$ solution of Oracle ▷ avg constraint w.r.t. w^t
  $w^{t+1} \leftarrow \mathbf{Update}(w^t, x^t)$
  ▷ decrease weight of constraints oversatisfied by x^t
$\bar{x} = \mathbf{avg}(x_1, \cdots x_T)$

MWU Update Rule:
$w_e^{t+1} := w_e^t \left(1 - \varepsilon/\phi(A_e x^t - b_e)\right)$

MWU to solve LP

Algorithm:

- Instead of solving for all the constraints, solve for a weighted average constraint.
- Take the solution
- The less a constraint is satisfied, the less weight it gets for the next iteration
- Repeat the above for T iterations
- Report the average solution found over all iterations.
- $T =$

CoveringLP($A_{n\times m}, c_m, b_n$)

Min $c^T x$

$$Ax \geq b$$
$$x \geq 0$$

Oracle($A_{n\times m}, c_m, b_n, p^t$)

Min $c^T x$

$$(w^t)^T Ax \geq (w^t)^T b$$
$$x \geq 0$$

$w^1 \leftarrow (1, \cdots, 1)$ ⊳ uniform weights
For $t = 1, t \leq T$ **do** ⊳ T iterations
  $x^t \leftarrow$ solution of Oracle ⊳ avg constraint w.r.t. w^t
  $w^{t+1} \leftarrow \textbf{Update}(w^t, x^t)$
  ⊳ decrease weight of constraints oversatisfied by x^t
$\bar{x} = \textbf{avg}(x_1, \cdots x_T)$

MWU Update Rule:

$$w_e^{t+1} := w_e^t \left(1 - \varepsilon/\phi(A_e x^t - b_e)\right)$$

$$\forall i: \quad A_e \bar{x} \geq b_e - \varepsilon$$

MWU to solve LP

Algorithm:

- Instead of solving for all the constraints, solve for a weighted average constraint.
- Take the solution
- The less a constraint is satisfied, the less weight it gets for the next iteration
- Repeat the above for T iterations
- Report the average solution found over all iterations.
- $T = O(\phi \log n / \epsilon^2)$

$w^1 \leftarrow (1, \cdots, 1)$ ⊳ uniform weights
For $t = 1, t \leq T$ **do** ⊳ T iterations
  $x^t \leftarrow$ solution of Oracle ⊳ avg constraint w.r.t. w^t
  $w^{t+1} \leftarrow \mathbf{Update}(w^t, x^t)$
  ⊳ decrease weight of constraints oversatisfied by x^t
$\bar{x} = \mathbf{avg}(x_1, \cdots x_T)$

CoveringLP($A_{n \times m}, c_m, b_n$)

Min $c^T x$
$$Ax \geq b$$
$$x \geq 0$$

Oracle($A_{n \times m}, c_m, b_n, p^t$)

Min $c^T x$
$$(w^t)^T A x \geq (w^t)^T b$$
$$x \geq 0$$

MWU Update Rule:
$$w_e^{t+1} := w_e^t \left(1 - \boldsymbol{\varepsilon}/\boldsymbol{\phi}(A_e x^t - b_e)\right)$$

$$\forall i: \quad A_e \bar{x} \geq b_e - \varepsilon$$

MWU to solve LP

Algorithm:

- Instead of solving for all the constraints, solve for a weighted average constraint.
- Take the solution
- The less a constraint is satisfied, the less weight it gets for the next iteration
- Repeat the above for T iterations
- Report the average solution found over all iterations.
- $T = O(\phi \log n / \epsilon^2)$

CoveringLP($A_{n\times m}, c_m, b_n$)

Min $c^T x$

$$Ax \geq b$$
$$x \geq 0$$

Oracle($A_{n\times m}, c_m, b_n, p^t$)

Min $c^T x$

$$(w^t)^T Ax \geq (w^t)^T b$$
$$x \geq 0$$

$w^1 \leftarrow (1, \cdots, 1)$ ⊳ uniform weights
For $t = 1, t \leq T$ **do** ⊳ T iterations
  $x^t \leftarrow$ solution of Oracle ⊳ avg constraint w.r.t. w^t
  $w^{t+1} \leftarrow \mathbf{Update}(w^t, x^t)$
  ⊳ decrease weight of constraints oversatisfied by x^t
$\bar{x} = \mathbf{avg}(x_1, \cdots x_T)$

MWU Update Rule:
$w_e^{t+1} := w_e^t \left(1 - \boldsymbol{\varepsilon}/\boldsymbol{\phi}(A_e x^t - b_e)\right)$

$\forall i:\quad A_e \bar{x} \geq b_e - \varepsilon$

$\forall i, t: -\phi \leq A_e x^t - b_e \leq \phi$

The Plan

- The Multiplicative Weight Update framework
  - **MWU for the Set Cover**
  - The average constraint: Oracle
- Implement MWU Oracle Naively in the streaming
  - $O(\frac{k \log n}{\epsilon^2})$ passes
- Reducing the number of passes to logarithmic
  - Reducing Width via Extended Set System
  - Fractional Max *k*-Cover
- Reducing the number of passes to a constant
  - Running several rounds of MWU together by sampling in advance

Multiplicative Weight Update (Set Cover)

<u>SET-COVER LP</u>($\mathcal{F}, \mathcal{U}$):

Min $\sum_{S \in \mathcal{F}} x_S$

s.t. $\sum_{S : e \in S} x_S \geq 1 \qquad \forall e \in \mathcal{U}$

$x_S \geq 0 \qquad \forall S \in \mathcal{F}$

Multiplicative Weight Update (Set Cover)

Feasibility-SET-COVER LP($\mathcal{F}$, $\mathcal{U}$, k)

$$\sum_{S \in \mathcal{F}} x_S \leq k$$

s.t.

$$\sum_{S : e \in S} x_S \geq 1 \qquad \forall e \in \mathcal{U}$$

$$x_S \geq 0 \qquad \forall S \in \mathcal{F}$$

Multiplicative Weight Update (Set Cover)

Feasibility-SET-COVER LP$(\mathcal{F}, \mathcal{U}, k)$

$$\sum_{S \in \mathcal{F}} x_S \leq k$$

s.t. $\sum_{S: e \in S} x_S \geq 1 \quad \forall e \in \mathcal{U}$

$x_S \geq 0 \quad \forall S \in \mathcal{F}$

Assign weight w_e to each element *e* (initially one)

Solve the *weighted* average constraint approximately!

Multiplicative Weight Update (Set Cover)

Feasibility-SET-COVER LP$(\mathcal{F}, \mathcal{U}, k)$

$$\sum_{S\in\mathcal{F}} x_S \le k$$

$$\sum_{e\in\mathcal{U}} w_e \sum_{e\in S} x_S \ge \sum_{e\in\mathcal{U}} w_e$$

$$x_S \ge 0 \qquad \forall S \in \mathcal{F}$$

Assign weight w_e to each element *e* (initially one)

Solve the *weighted* average constraint approximately!

$$\sum_{e\in\mathcal{U}} w_e \sum_{e\in S} x_S \ge \sum_{e\in\mathcal{U}} w_e$$

$$\sum_{S\in\mathcal{F}} x_S \sum_{e\in S} w_e \ge \sum_{e\in\mathcal{U}} w_e$$

$$\sum_{S\in\mathcal{F}} x_S w_S \ge \sum_{e\in\mathcal{U}} w_e$$

Define $w_S := \sum_{e\in S} w_e$

By *normalizing* weight vector w (prob. vector p):

$$\sum_{S\in\mathcal{F}} x_S p_S \ge 1$$

Multiplicative Weight Update (Set Cover)

Oracle($\mathcal{F}, \mathcal{U}, k, p$)

$$\sum_{S \in \mathcal{F}} x_S \leq k$$

$$\sum_{S \in \mathcal{F}} x_S p_S \geq 1$$

$$x_S \geq 0 \qquad \forall S \in \mathcal{F}$$

Assign weight w_e to each element e (initially one)

Solve the *weighted* average constraint approximately!

Multiplicative Weight Update (Set Cover)

Oracle$(\mathcal{F}, \mathcal{U}, k, p)$

$$\sum_{S\in\mathcal{F}} x_S \le k$$

$$\sum_{S\in\mathcal{F}} x_S p_S \ge 1 - \varepsilon$$

$$x_S \ge 0 \qquad \forall S \in \mathcal{F}$$

$$-\phi \le \sum_{S: e\in S} x_S - 1 \le \phi \;\; \forall e \in \mathcal{U}$$

Width of oracle

Bounding the max number of times an element gets covered

Assign weight w_e to each element e (initially one)

T times

Solve the *weighted* average constraint approx. w.r.t $p^t (\propto w^t)$: $\boldsymbol{x^t}$

Update the prob vector

$$\boldsymbol{p_e^{t+1} \coloneqq p_e^t\left(1 - O(\varepsilon) \times \left(\sum x_S^t - 1\right)\right)}$$

MWU Theorem. After $T = O(\frac{\phi \log n}{\varepsilon^2})$ rounds, $\bar{x} = \frac{1}{T}(x^1 + \cdots + x^t)$ is an ε-feasible solution.

$$\sum_{S: e\in S} x_S \ge 1 - \varepsilon \qquad \forall e \in \mathcal{U}$$

Finally, we can then pick $k(1+\epsilon)$ sets to cover all the elements!

The Plan

- The Multiplicative Weight Update framework
  - MWU for the Set Cover
  - **The average constraint: Oracle**
- Implement MWU Oracle Naively in the streaming
  - $O(\frac{k \log n}{\epsilon^2})$ passes
- Reducing the number of passes to logarithmic
  - Reducing Width via Extended Set System
  - Fractional Max *k*-Cover
- Reducing the number of passes to a constant
  - Running several rounds of MWU together by sampling in advance

The Oracle

Given: a probability vector p on the elements, and k

Goal: pick (fractionally) k sets by assigning values to x_S such that

1. *The total probability* (weight) of the sets in the solution is *maximized*, i.e., at least $(1-\varepsilon)$, where
   - probability of a set is the sum of the probability of its elements, i.e., $p_S = \sum_{e \in S} p_e$
2. The width (total number of times any *element is covered)* is *small*.

Oracle$(\mathcal{F}, \mathcal{U}, k, p)$

$$\sum_{S \in \mathcal{F}} x_S \le k$$

$$\sum_{S \in \mathcal{F}} x_S p_S \ge 1 - \varepsilon$$

$$x_S \ge 0 \qquad \forall S \in \mathcal{F}$$

$$-\phi \le \sum_{S: e \in S} x_S - 1 \le \phi \qquad \forall e \in \mathcal{U}$$

Initial plan:

- solve the Oracle in one pass and low space,
- gives an algorithm for set cover with T passes and low space.

The Plan

- The Multiplicative Weight Update framework
  - MWU for the Set Cover
  - The average constraint: Oracle
- **Implement MWU Oracle Naively in the streaming**
  - $O(\frac{k \log n}{\epsilon^2})$ passes
- Reducing the number of passes to logarithmic
  - Reducing Width via Extended Set System
  - Fractional Max *k*-Cover
- Reducing the number of passes to a constant
  - Running several rounds of MWU together by sampling in advance

Implementing MWU in Stream (I)

- Naïve solution for the oracle: → $x_S = \begin{cases} k & \text{If S is the **heaviest** set,} \\ 0 & \text{Otherwise.} \end{cases}$
- **Width** (the number of times an element is covered) is trivially k
- The number of required rounds to obtain $(1+\epsilon)$-approximation is $O(\frac{k \log n}{\varepsilon^2})$
- **Streaming:** find the **heaviest** set w.r.t $\boldsymbol{p}$ in a single pass over the stream

Performance

$(1+\varepsilon)$-approximation
$O({}^{k \log n}/_{\varepsilon^2})$ passes
$\tilde{O}(n)$ space

Oracle$(\mathcal{F}, \mathcal{U}, k, p)$

$$\sum_{S \in \mathcal{F}} x_S \le k$$

$$\sum_{S \in \mathcal{F}} x_S p_S \ge 1 - \varepsilon$$

$$x_S \ge 0 \qquad \forall S \in \mathcal{F}$$

$$-\phi \le \sum_{S: e \in S} x_S - 1 \le \phi \qquad \forall e \in \mathcal{U}$$

Challenge:
Is it possible to find a solution to the oracle with **smaller width**?

No, simply all sets may contain a **designated element e** and hence the width of any solution to the oracle is always k no matter how the solution is picked.

The Plan

- The Multiplicative Weight Update framework
  - MWU for the Set Cover
  - The average constraint: Oracle
- Implement MWU Oracle Naively in the streaming

$(1+\varepsilon)$-appx	$O(^{k \log n}/_{\varepsilon^2})$-pass	$\tilde{O}(n)$-space

- **Reducing the number of passes to logarithmic**
  - Reducing Width via Extended Set System
  - Fractional Max *k*-Cover
- Reducing the number of passes to a constant
  - Running several rounds of MWU together by sampling in advance

The Plan

- The Multiplicative Weight Update framework
  - MWU for the Set Cover
  - The average constraint: Oracle
- Implement MWU Oracle Naively in the streaming

$(1+\varepsilon)$-appx	$O(^{k \log n}/_{\varepsilon^2})$-pass	$\tilde{O}(n)$-space

- **Reducing the number of passes to logarithmic**
  - **Reducing Width via Extended Set System**
  - Fractional Max *k*-Cover
- Reducing the number of passes to a constant
  - Running several rounds of MWU together by sampling in advance

Extended Set System

Challenge:
Is it possible to find a solution to the oracle in *set system* $(\mathcal{U}, \mathcal{F})$ with **smaller width**?

No, simply all sets may contain a **designated element *e*** and hence the width of any solution to the oracle is always k no matter how the solution is picked.

Different Set System?

Extended Set System of $\mathcal{F}$:
The set system $(\mathcal{U}, \check{\mathcal{F}})$ (*extension of* $\mathcal{F}$) is the collection containing all subsets of sets in $\mathcal{F}$.

$$\mathcal{F} = \{\{1,2,3\}, \{3,4,5\}, \{2,6\}\}$$

$$\check{\mathcal{F}} = \{$$
$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$$
$$\{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{4,5\}, \{2,6\}$$
$$\{1,2,3\}, \{3,4,5\}$$
$$\}$$

Extended Set System

✓ The size of an optimal cover in both set systems are the same.

Different Set System?

Extended Set System of $\mathcal{F}$:
The set system $(\mathcal{U}, \check{\mathcal{F}})$ (*extension of* $\mathcal{F}$) is the collection containing all subsets of sets in $\mathcal{F}$.

$$\mathcal{F} = \{\{1,2,3\}, \{3,4,5\}, \{2,6\}\}$$

$$\check{\mathcal{F}} = \{$$
$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$$
$$\{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{4,5\}, \{2,6\}$$
$$\{1,2,3\}, \{3,4,5\}$$
$$\}$$

Extended Set System

✓ The size of an optimal cover in both set systems are the same.

Different Set System?

Extended Set System of $\mathcal{F}$:
The set system $(\mathcal{U}, \check{\mathcal{F}})$ (*extension of* $\mathcal{F}$) is the collection containing all subsets of sets in $\mathcal{F}$.

$$\mathcal{F} = \{\{1,2,3\}, \{3,4,5\}, \{2,6\}\}$$

$$\check{\mathcal{F}} = \{$$
$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$$
$$\{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{4,5\}, \{2,6\}$$
$$\{1,2,3\}, \{3,4,5\}$$
$$\}$$

Extended Set System

- ✓ The size of an optimal cover in both set systems are the same.
- ✓ We can easily find an optimal solution with width **one** in the extended set system $\check{\mathcal{F}}$

Different Set System?

Extended Set System of $\mathcal{F}$:
The set system $(\mathcal{U}, \check{\mathcal{F}})$ (*extension of $\mathcal{F}$*) is the collection containing all subsets of sets in $\mathcal{F}$.

$$\mathcal{F} = \{\{1,2,3\}, \{3,4,5\}, \{2,6\}\}$$

$$\check{\mathcal{F}} = \{$$
$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$$
$$\{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{4,5\}, \{2,6\}$$
$$\{1,2,3\}, \{3,4,5\}$$
$$\}$$

Extended Set System

- ✓ The size of an optimal cover in both set systems are the same.
- ✓ We can easily find an optimal solution with width **one** in the extended set system $\check{\mathcal{F}}$

Different Set System?

Extended Set System of $\mathcal{F}$:
The set system $(\mathcal{U}, \check{\mathcal{F}})$ (*extension of* $\mathcal{F}$) is the collection containing all subsets of sets in $\mathcal{F}$.

$$\mathcal{F} = \{\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}, \{3,4,5\}, \{2,6\}\}$$

$$\check{\mathcal{F}} = \{$$
$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$$
$$\{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{4,5\}, \{2,6\}$$
$$\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}, \{3,4,5\}$$
$$\}$$

Extended Set System

- ✓ The size of an optimal cover in both set systems are the same.
- ✓ We can easily find an optimal solution with width **one** in the extended set system $\check{\mathcal{F}}$

Different Set System?

Extended Set System of $\mathcal{F}$:
The set system $(\mathcal{U}, \check{\mathcal{F}})$ (*extension of $\mathcal{F}$*) is the collection containing all subsets of sets in $\mathcal{F}$.

$$\mathcal{F} = \{\{1,2,3\}, \{\mathbf{3}, \mathbf{4}, \mathbf{5}\}, \{2,6\}\}$$

$$\check{\mathcal{F}} = \{$$
$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$$
$$\{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{\mathbf{4}, \mathbf{5}\}, \{2,6\}$$
$$\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}, \{3,4,5\}$$
$$\}$$

Extended Set System

- ✓ The size of an optimal cover in both set systems are the same.
- ✓ We can easily find an optimal solution with width **one** in the extended set system $\check{\mathcal{F}}$

Different Set System?

> **Extended Set System of $\mathcal{F}$:**
> The set system $(\mathcal{U}, \check{\mathcal{F}})$ (*extension of* $\mathcal{F}$) is the collection containing all subsets of sets in $\mathcal{F}$.

$$\mathcal{F} = \{\{1,2,3\},\{3,4,5\},\{\mathbf{2},\mathbf{6}\}\}$$

$$\check{\mathcal{F}} = \{$$
$$\{1\},\{2\},\{3\},\{4\},\{5\},\{\mathbf{6}\}$$
$$\{1,2\},\{1,3\},\{2,3\},\{3,4\},\{3,5\},\{\mathbf{4},\mathbf{5}\},\{2,6\}$$
$$\{\mathbf{1},\mathbf{2},\mathbf{3}\},\{3,4,5\}$$
$$\}$$

Extended Set System

✓ The size of an optimal cover in both set systems are the same.

✓ We can easily find an optimal solution with width **one** in the extended set system $\check{\mathcal{F}}$

- Idea: Pruning the cover

Different Set System?

Extended Set System of $\mathcal{F}$:
The set system $(\mathcal{U}, \check{\mathcal{F}})$ (*extension of* $\mathcal{F}$) is the collection containing all subsets of sets in $\mathcal{F}$.

$$\mathcal{F} = \{\{1,2,3\},\{3,4,5\},\{\mathbf{2},\mathbf{6}\}\}$$

$$\check{\mathcal{F}} = \{$$
$$\{1\},\{2\},\{3\},\{4\},\{5\},\{\mathbf{6}\}$$
$$\{1,2\},\{1,3\},\{2,3\},\{3,4\},\{3,5\},\{\mathbf{4},\mathbf{5}\},\{2,6\}$$
$$\{\mathbf{1},\mathbf{2},\mathbf{3}\},\{3,4,5\}$$
$$\}$$

Extended Set System

- ✓ The size of an optimal cover in both set systems are the same.
- ✓ We can easily find an optimal solution with width **one** in the extended set system $\check{\mathcal{F}}$
  - Idea: Pruning the cover

Different Set System?

Extended Set System of $\mathcal{F}$:
The set system $(\mathcal{U}, \check{\mathcal{F}})$ (*extension of $\mathcal{F}$*) is the collection containing all subsets of sets in $\mathcal{F}$.

- ➢ Extended Set System has exponentially many sets
  - Work with the original set system,
  - Solve the oracle on $\mathcal{F}$ but and convert it to a solution for $\check{\mathcal{F}}$

$$\mathcal{F} = \{\{1,2,3\}, \{3,4,5\}, \{\mathbf{2}, \mathbf{6}\}\}$$

$$\check{\mathcal{F}} = \{$$
$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{\mathbf{6}\}$$
$$\{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{\mathbf{4}, \mathbf{5}\}, \{2,6\}$$
$$\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}, \{3,4,5\}$$
$$\}$$

Implementing MWU in Stream (II)

- We want to solve the oracle for $(\mathcal{U}, \check{\mathcal{F}})$
  - Find some solution for the oracle $(\mathcal{U}, \mathcal{F})$, ➡ e.g., $x_S = \begin{cases} 1 & \text{If S is one of the } \textbf{k heaviest} \text{ set,} \\ 0 & \text{Otherwise.} \end{cases}$
  - Prune it to get a solution for $(\mathcal{U}, \check{\mathcal{F}})$
    - ✔ Obtains width = 1
    - ✖ The average constraint may not be satisfied any more!
- Instead find a solution that maximizes coverage
  - ✔ Coverage remains unchanged after pruning
  - There is a cover of size k,
  - ✔ The solution of maximum k-coverage satisfies the average constraint of the set cover too; even after the pruning: $\sum_{S \in \mathcal{F}} x_S p_S \geq \sum_{e \in \mathcal{U}} p_e = 1$

Oracle$(\mathcal{F}, \mathcal{U}, k, p)$

$$\sum_{S \in \mathcal{F}} x_S \leq k$$

$$\sum_{S \in \mathcal{F}} x_S p_S \geq 1 - \varepsilon$$

$$x_S \geq 0 \qquad \forall S \in \mathcal{F}$$

$$-1 \leq \sum_{S: e \in S} x_S - 1 \leq 1 \qquad \forall e \in \mathcal{U}$$

Next Goal:
Given a set system $(\mathcal{U}, \mathcal{F})$, and a parameter k, solve the (weighted) fractional Max k-Cover in one pass

The Plan

- The Multiplicative Weight Update framework
  - MWU for the Set Cover
  - The average constraint: Oracle
- Implement MWU Oracle Naively in the streaming

$(1+\varepsilon)$-appx	$O(^{k \log n}/_{\varepsilon^2})$-pass	$\tilde{O}(n)$-space

- **Reducing the number of passes to logarithmic**
  - Reducing Width via Extended Set System
  - **Fractional Max *k*-Cover**
- Reducing the number of passes to a constant

Max k-Cover Problem

Input: a collection $\mathcal{F}$ of sets $S_1, \ldots, S_m$

Each $S \subseteq \mathcal{U} = \{1, \ldots, n\}$

Output: k sets of $\mathcal{F}$ such that:

Maximizes the total coverage;

$|\bigcup_{S \in \mathcal{C}} S|$

Complexity:

- NP-hard
- Greedy: $(1 - \frac{1}{e})$-approximation
- One pass $(1 - \varepsilon)$-approx. using $\tilde{O}(m/\varepsilon^2)$ space [MV17], [BEM17]

Fractional Max *k*-Cover

Max-Cover-LP$(\mathcal{F}, \mathcal{U}, k)$

Max. $\sum_{e \in \mathcal{U}} z_e$

s.t.

$$\begin{aligned}
\sum_{S \in \mathcal{F}} x_S &\leq k & \\
\sum_{S: e \in S} x_S &\geq z_e & \forall e \in \mathcal{U} \\
x_S &\geq 0 & \forall S \in \mathcal{F} \\
z_e &\leq 1 & \forall e \in \mathcal{U}
\end{aligned}$$

Weighted Max k-Cover Problem

Input: a collection $\mathcal{F}$ of sets $S_1, \ldots, S_m$

Each $S \subseteq \mathcal{U} = \{1, \ldots, n\}$

Output: *k* sets of $\mathcal{F}$ such that:

Maximizes the total coverage;

$|\cup_{S \in \mathcal{C}} S|$

Complexity:

- NP-hard
- Greedy: $(1 - \frac{1}{e})$-approximation
- One pass $(1 - \varepsilon)$-approx. using $\tilde{O}(m/\varepsilon^2)$ space [MV17], [BEM17]

Fractional (Weighted) Max *k*-Cover

Max-Cover-LP$(\mathcal{F}, \mathcal{U}, k, p)$

Max. $\sum_{e \in \mathcal{U}} p_e z_e$

s.t.

$$\sum_{S \in \mathcal{F}} x_S \leq k$$
$$\sum_{S : e \in S} x_S \geq z_e \quad \forall e \in \mathcal{U}$$
$$x_S \geq 0 \quad \forall S \in \mathcal{F}$$
$$z_e \leq 1 \quad \forall e \in \mathcal{U}$$

Fractional Max k-Cover in One Pass

- **Component I (Element Sampling):**
  1. Sample $\tilde{O}(\frac{k}{\varepsilon^2})$ elements in U' according to $\boldsymbol{p}$.
  2. In one pass over the stream: Store $\mathcal{F}'$, the intersection of all sets in $\mathcal{F}$ with U'
  3. Return the best *k*-cover of the sampled elements.
  - w.h.p. the constructed cover is a $(1-\varepsilon)$-approximate solution of the main instance.
  - Required space: $\tilde{O}(mk/\varepsilon^2)$
- **Component II (Covering Common Elements):**
  - In the preprocessing step, pick x^{cmn}=$\left\langle\frac{\varepsilon k}{m}, \ldots, \frac{\varepsilon k}{m}\right\rangle$
  - All frequently occurring elements will be covered.
  - We can focus on elements with degree $\leq \frac{m}{\varepsilon k}$
  - Required space: $\tilde{O}\left(\frac{m}{\varepsilon k} \times \frac{k}{\varepsilon^2}\right) = \tilde{O}(m/\varepsilon^3)$

The pruning

We have:

- Solution $\vec{x}$ on the original set system $(U, \mathcal{F})$
- The coverage $y_e := \sum_{S \ni e} x_S$ of every element by the solution of the original set system $\vec{x}$ can be computed in one pass.

We need:

- Convert $\vec{x}$ to a solution $\overrightarrow{x'}$ on the extended set system $(U, \check{\mathcal{F}})$ so that $\overrightarrow{x'}$ can be averaged in the end of the T iterations.
- The coverage $y_e' := \sum_{S \ni e} x_S{}'$ by the solution $\overrightarrow{x'}$ to update the weights of MWU
  - $\boldsymbol{p_e^{t+1}} := \boldsymbol{p_e^t}\left(\boldsymbol{1 - O(\varepsilon) \times (y_e{}' - 1)}\right)$

➢ **The Pruning:** needs to be done fractionally.

<u>**Lemma:**</u> There exists a polynomial time algorithm to prune the fractional solution $\vec{x}$ of the maximum coverage on $(U, \mathcal{F})$ to get a solution $\overrightarrow{x'}$ of $(U, \check{\mathcal{F}})$ s.t. the coverage of every element is capped by 1, i.e., $y_e{}' = \mathrm{Min}(y_e, 1)$.

Implementing MWU in Stream (II)

- Solve fractional Max k Cover in one pass find $\vec{x}$ and in one pass y_e
- Obtain $\overrightarrow{x'}$ and y'_e using the lemma.
- $\overrightarrow{x'}$ satisfies the average constraint.
- Update the probabilities according to y'_e
- width is 1
- The number of required rounds of MWU is $O(\frac{\log n}{\varepsilon^2})$

Performance

$(1+\epsilon)$-approximation
$O({}^{\log n}/{}_{\varepsilon^2})$ passes
$\tilde{O}({}^{m}/{}_{\varepsilon^3})$ space

Oracle$(\mathcal{F}, \mathcal{U}, k, p)$

$$\sum_{S\in\mathcal{F}} x_S \le k$$

$$\sum_{S\in\mathcal{F}} x_S p_S \ge 1-\varepsilon$$

$$x_S \ge 0 \quad \forall S \in \mathcal{F}$$

$$-1 \le \sum_{S:e\in S} x_S - 1 \le 1 \quad \forall e \in \mathcal{U}$$

Challenge:
Can we run several rounds of MWU in one pass of the streaming algorithm?

The Plan

- The Multiplicative Weight Update framework
  - MWU for the Set Cover
  - The average constraint: Oracle
- Implement MWU Oracle Naively in the streaming

$(1+\varepsilon)$-appx	$O({}^{k\log n}/_{\varepsilon^2})$-pass	$\tilde{O}(n)$-space

- Reducing the number of passes to logarithmic
  - Reducing Width via Extended Set System
  - Fractional Max *k*-Cover

$(1+\varepsilon)$-appx	$O({}^{\log n}/_{\varepsilon^2})$-pass	$\tilde{O}(m/\epsilon^3)$-space

- **Reducing the number of passes to a constant**
  - Running several rounds of MWU together by sampling in advance

Reducing the Number of Passes Further!

Perform several rounds of MWU in one pass

- × But probability distribution p changes over the iterations
- × Element sampling is done w.r.t. p

Key observation:
The probability vector p changes slowly.

Component I (Element Sampling):
Sample $\tilde{O}(\frac{k}{\varepsilon^2})$ elements according to $\boldsymbol{p}$.
Return the best k-cover of the sampled elements.

After ℓ rounds of MWU:

$p_e^{t+\ell} \leq p_e^t\big(1 + O(\varepsilon)\big)^\ell$

Setting $\ell = O(\frac{\log n}{\varepsilon^2 d})$ rounds, p_e increases at most by $n^{O(\frac{1}{\varepsilon d})}$

Reducing the Number of Passes Further!

Perform several rounds of MWU in one pass

- × But probability distribution p changes over the iterations
- × Element sampling is done w.r.t. p

Key observation:
The probability vector p changes slowly.

Component I (Element Sampling):
Sample $\tilde{O}(\frac{kn^{O(1/\varepsilon d)}}{\varepsilon^2})$ elements according to $\boldsymbol{p}$.
Return the best k-cover of the sampled elements.

Rejection Sampling: To adjust the probability p_e

Keep each sample w.p. $p_e^{t+\ell}/p_e^t n^{O(1/\varepsilon d)}$

After ℓ rounds of MWU:
$p_e^{t+\ell} \leq p_e^t\left(1 + O(\varepsilon)\right)^\ell$

Setting $\ell = O(\frac{\log n}{\varepsilon^2 d})$ rounds, p_e increases at most by $n^{O(\frac{1}{\varepsilon d})}$

To perform $O(\frac{\log n}{\varepsilon^2 d})$ rounds together

Reducing the Number of Passes Further!

Perform several rounds of MWU in one pass

- × But probability distribution p changes over the iterations
- × Element sampling is done w.r.t. p

Key observation:
The probability vector p changes slowly.

After ℓ rounds of MWU:

$$p_e^{t+\ell} \leq p_e^t \left(1 + O(\varepsilon)\right)^\ell$$

Setting $\ell = O(\frac{\log n}{\varepsilon^2 d})$ rounds, p_e increases at most by $n^{O(\frac{1}{\varepsilon d})}$

Component I (Element Sampling):
Sample $\tilde{O}(\frac{kn^{O(1/\varepsilon d)}}{\varepsilon^2})$ elements according to $\boldsymbol{p}$.
Return the best k-cover of the sampled elements.

+ **Rejection Sampling**: To adjust the probability p_e

→

Space *increases* by $n^{O(1/\varepsilon d)}$
#passes *decreases* by $O(\frac{\log n}{\varepsilon^2 d})$

Implementing MWU in Stream (II)

- Algorithm will go over d passes:
  - Sample $\tilde{O}(\frac{kn^{O(1/\varepsilon d)}}{\varepsilon^2})$ elements for each of the $O\left(\frac{\log n}{\epsilon^2 d}\right)$ rounds assigned to this pass.
  - In one pass find the projection of all sets on these sampled elements in $\tilde{O}\left(mn^{O(1/d\varepsilon)}\right)$ space. (this uses the common element component).
  - For each of the $O\left(\frac{\log n}{\epsilon^2 d}\right)$ rounds.
    - Adjust the samples properly.
    - Solve fractional Max k Cover find x_S
    - Update the probabilities for all the sampled elements
  - In one pass update the probabilities for all the elements.

Oracle$(\mathcal{F}, \mathcal{U}, k, p)$

$$\sum_{S\in\mathcal{F}} x_S \le k$$

$$\sum_{S\in\mathcal{F}} x_S p_S \ge 1-\varepsilon$$

$$x_S \ge 0 \qquad \forall S \in \mathcal{F}$$

$$-1 \le \sum_{S:e\in S} x_S - 1 \le 1 \quad \forall e \in \mathcal{U}$$

Performance

$(1+\varepsilon)$-approximation
$O(d)$ passes
$\tilde{O}(mn^{O(1/d\varepsilon)})$ space

The Plan

- The Multiplicative Weight Update framework
  - MWU for the Set Cover
  - The average constraint: Oracle
- Implement MWU Oracle Naively in the streaming

$(1+\varepsilon)$-appx	$O({}^{k \log n}/_{\varepsilon^2})$-pass	$\tilde{O}(n)$-space

- Reducing the number of passes to logarithmic
  - Reducing Width via Extended Set System
  - Fractional Max *k*-Cover

$(1+\varepsilon)$-appx	$O({}^{\log n}/_{\varepsilon^2})$-pass	$\tilde{O}(m/\epsilon^3)$-space

- Reducing the number of passes to a constant
  - Running several rounds of MWU together by sampling in advance

$(1+\varepsilon)$-appx	$O(p)$-pass	$\tilde{O}(mn^{O(1/d\varepsilon)})$-space

Summary

- Considered MWU for solving fractional-Set Cover
  - One pass for each of the $O(\frac{\phi \log n}{\epsilon^2})$ iterations.
  - Trivial solution gets $\phi = k$ giving $O(\frac{k \log n}{\epsilon^2})$
  - No way to reduce the width to smaller than k.

Performance

$(1+\varepsilon)$-approximation
$O(k \log n / \epsilon^2)$ passes
$\tilde{O}(n)$ space

- Change the set system to extended set system.
  - Solution remains the same.
  - Goal changes to weighted maximum coverage that is preserved under the pruning.
  - Obtain $\phi = 1$ giving $O(\frac{\log n}{\epsilon^2})$ pass algorithm

Performance

$(1+\epsilon)$-approximation
$O({}^{\log n}/_{\varepsilon^2})$ passes
$\tilde{O}({}^{m}/_{\varepsilon^3})$ space

- Run several rounds of MWU together
  - The probabilities change slowly over iterations.
  - Sample more elements in advance and adjust the probability.
  - Get constant pass algorithm.

Performance

$(1+\varepsilon)$-approximation
$O(d)$ passes
$\tilde{O}(mn^{O(1/d\varepsilon)})$ space

Open Questions

- **Open Questions:**
  1. Better bound for general covering/packing LP?
  2. Any constant pass polylog-approximation algorithm for Weighted Set Cover with $o(mn)$ space ?
  3. Optimal number of passes for O(log n)-approx. Set Cover?
     I. Best Upper Bound: O(log n)-pass
     II. Best Lower Bound: $\Omega(\frac{\log n}{\log\log n})$-pass [CW16]

Thank You