Lecture 4

TTIC 41000: Algorithms for Massive Data

Toyota Technological Institute at Chicago

Spring 2021

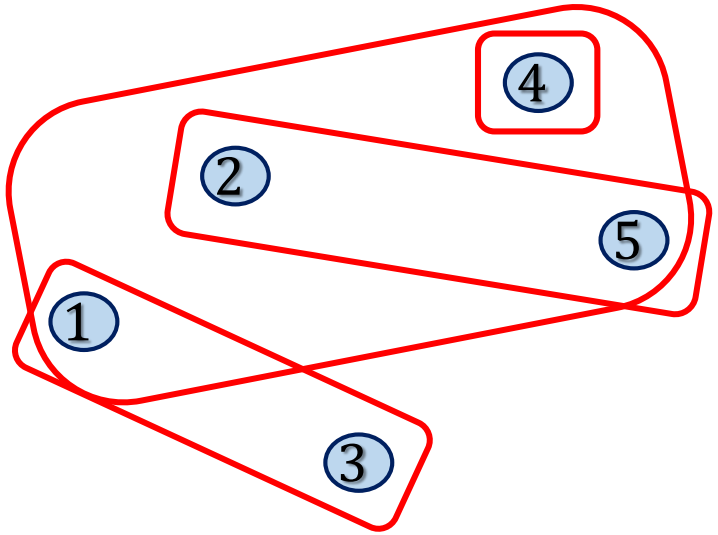
Instructor: Sepideh Mahabadi

This Lecture

- ❑ Multi-pass Algorithm for Set Cover
- ❑ Fractional Set Cover using MWU

Set Cover Problem

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

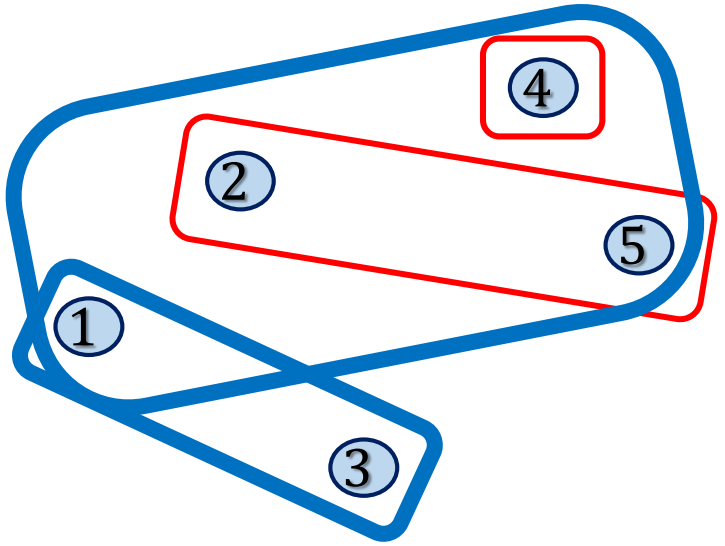


Set Cover Problem

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

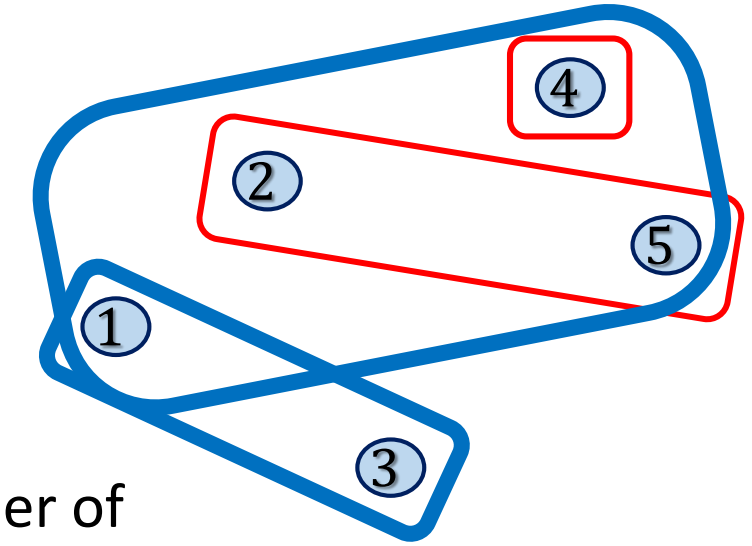


Set Cover Problem

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
  - Pick the set that covers maximum number of uncovered elements
- Can't do better unless **P=NP**
  [LY91][RS97][Fei98][AMS06][DS14]

Set Cover in Massive Data Models

- Studied in the massive data models
  - A classic optimization problem
  - Application in “Big Data”: Clustering, Topic Coverage
  - The streaming setting: [ER’14, DIMV’14, CW’16, HIMV’16, AKL’16, A’17, BEM’17, IMRUVY’17]

Goal: *“solve set cover in **the streaming model**”*

- Model:
  - Sequential access to $S_1, S_2, \dots, S_m$
  - One (or few) passes, sublinear (i.e., $o(mn)$) storage
  - (Hopefully) decent approximation factor

Naïve Algorithm

Naïve implementations of the greedy algorithm

1. In one pass, keep all the data and run greedy and the end of the stream

2. In each pass over the stream, keep track of the set that covers maximum number of uncovered elements.

What it gives

Algorithms	Approximation	Passes	Space	Type
Greedy Alg	$O(\log n)$	1	$O(mn)$	Deterministic
	$O(\log n)$	n	$O(n)$	Deterministic

n = number of *elements*

m = *number of sets.*

Better Greedy

For $T = 2^i$ where $i = \log n$ to 0

- In one pass, pick any set that covers at least T yet-uncovered elements.

- $\log n$ passes
- Space $O(n)$
- Exercise: gives $O(\log n)$ approximation

What it gives

Algorithms	Approximation	Passes	Space	Type
Greedy Alg	$O(\log n)$	1	$O(mn)$	Deterministic
	$O(\log n)$	n	$O(n)$	Deterministic
[GS'09]	$O(\log n)$	$O(\log n)$	$O(n \log n)$	Deterministic

$n =$ number of *elements*

$m =$ *number of sets.*

A Sampling-based algorithm

1. Two simple components used for coverage problems in massive data models.
   - Set Sampling
   - Element Sampling
2. The algorithm overview

➢ Lets assume we know k the value of the optimal solution (min set cover size)

Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, elements with degree at least $\frac{m}{\ell}$ are covered in expectation.

Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

Using Chernoff

Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

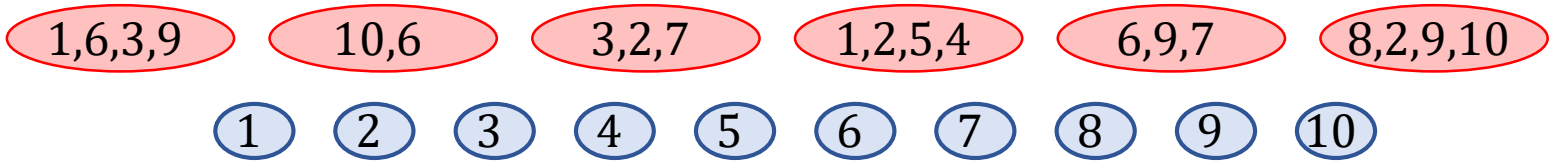
How we use the lemma: set $\ell \approx O(k)$

Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

$\ell = 2$

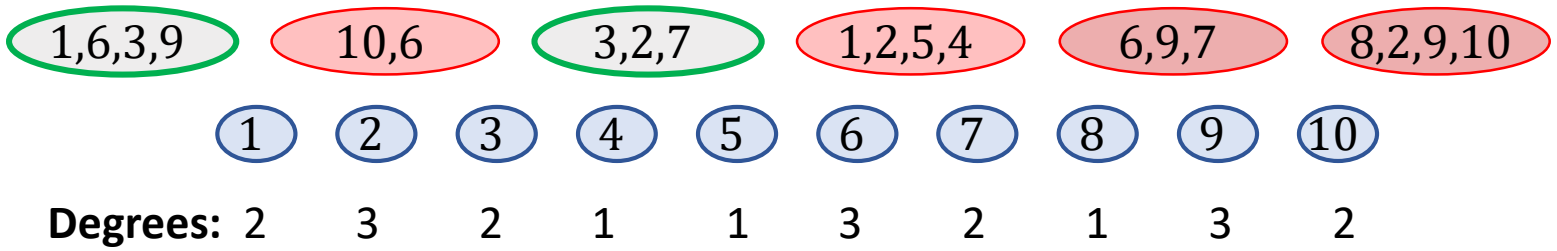


Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

$\ell = 2$

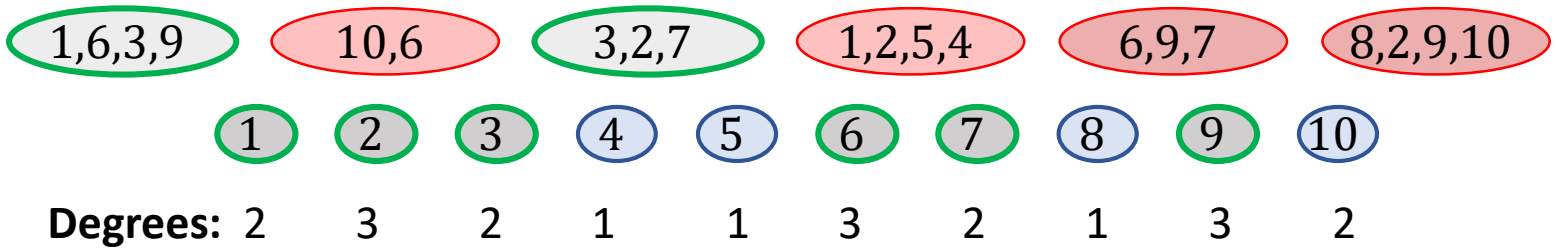


Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

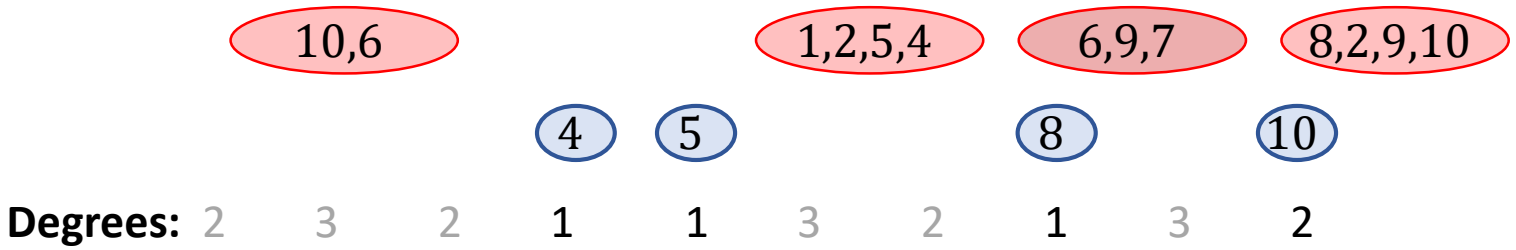
$\ell = 2$



Component I: set sampling

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

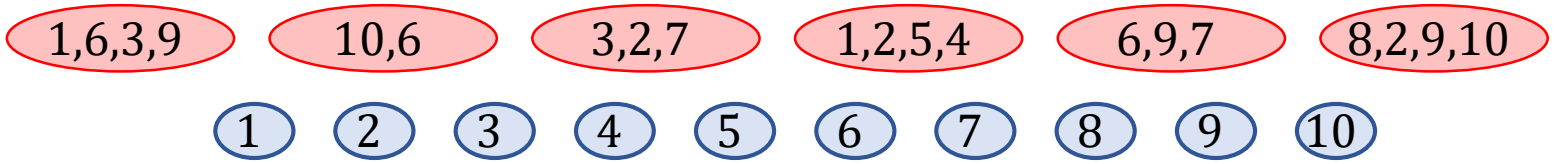


Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.

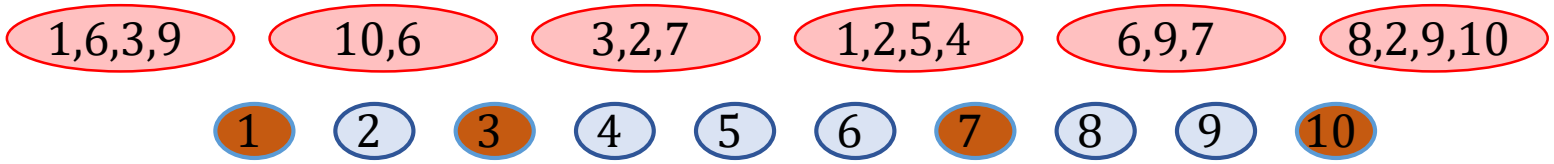
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



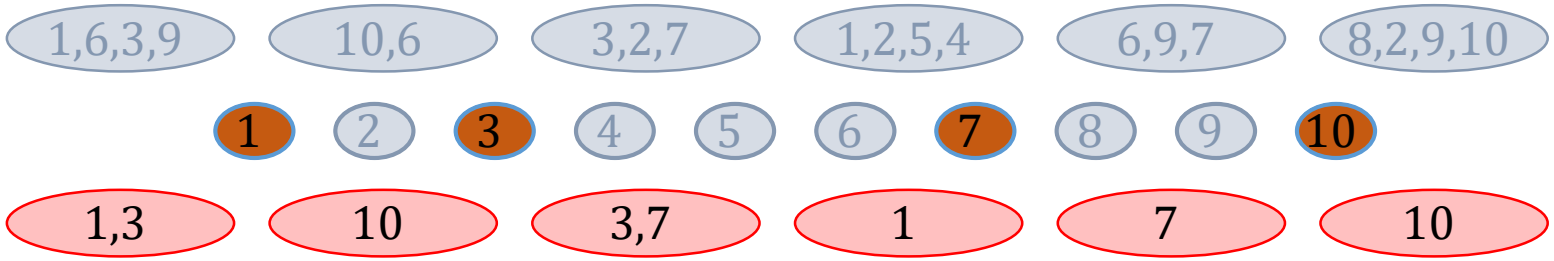
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



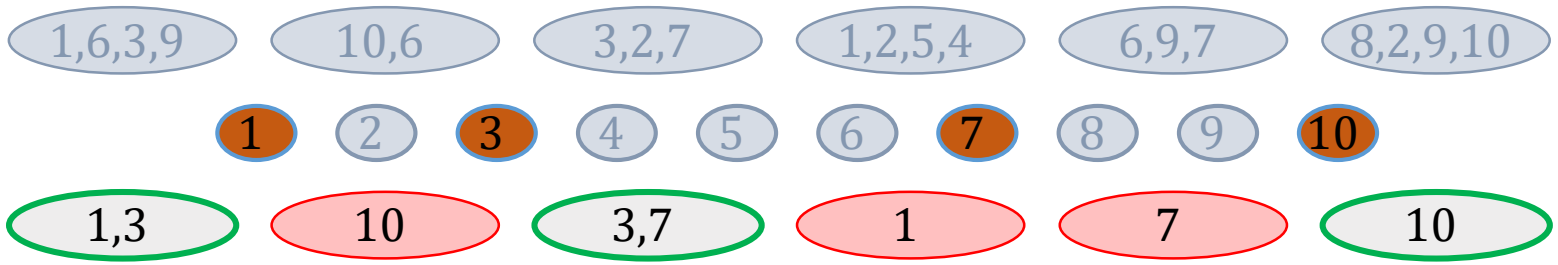
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



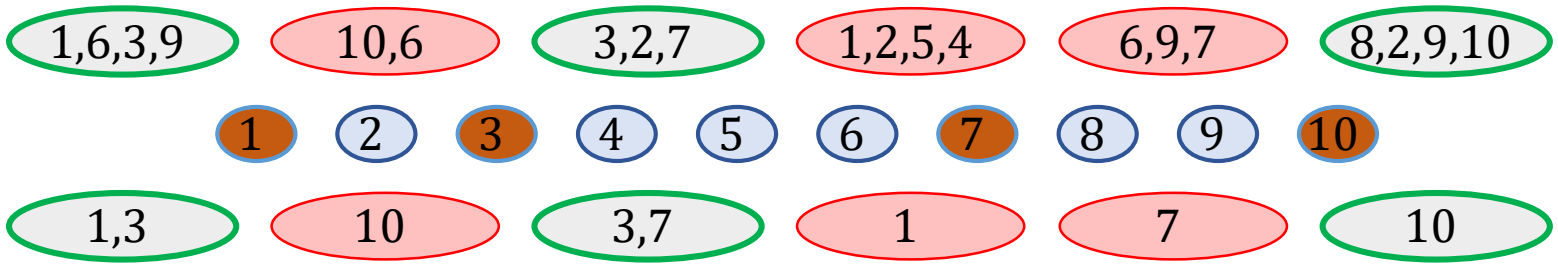
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



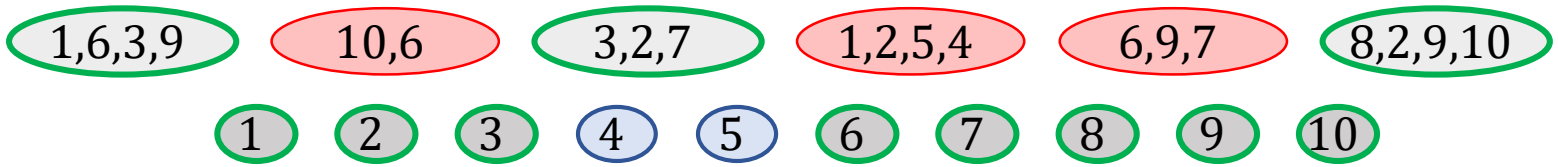
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



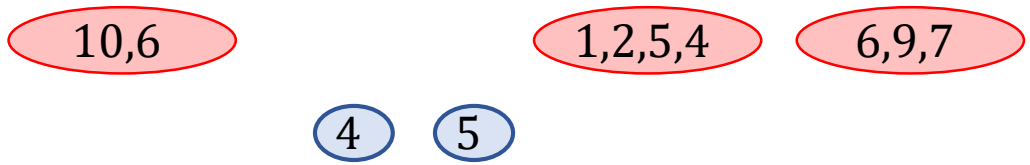
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



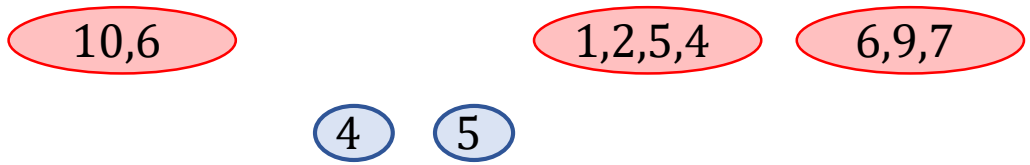
Component II: element sampling

Element Sampling: Sample a few elements and solve the set cover for the sampled elements.



Component II: element sampling

Element Sampling: Sampling $\Theta(\frac{\rho k \log m}{\epsilon})$ elements uniformly at random and finding a ρ-approximate cover for the sampled elements, will cover $(1-\epsilon)$ fraction of the original elements w.h.p.



Component II: element sampling

Element Sampling: Sampling $\Theta(\frac{\rho k \log m}{\epsilon})$ elements uniformly at random and finding a ρ-approximate cover for the sampled elements, will cover $(1-\epsilon)$ fraction of the original elements w.h.p.

$\boldsymbol{\rho}$ is the approximation factor of the **offline algorithm** we use

- $\log n$ if we want polynomial time algorithm
- 1 if we only care about the space usage of the algorithm

Component II: element sampling

Element Sampling: Sampling $\Theta(\frac{\rho k \log m}{\epsilon})$ elements uniformly at random and finding a ρ-approximate cover for the sampled elements, will cover $(1-\epsilon)$ fraction of the original elements w.h.p.

- Size of the optimal solution is k
- There are $m^{\rho k}$ subcollection of sets $\mathcal{F}' \subseteq \mathcal{F}$
- Take one $\mathcal{F}'$ suppose it covers less than $(1-\epsilon)$ fraction of the elements
- What is the probability that it becomes a cover after the sampling?
- The probability that we don't sample of the at least ϵn elements.
- $(1-\epsilon)^{\frac{c\rho k \log m}{\epsilon}} \approx \left(\frac{1}{e}\right)^{c\rho k \log m} = \left(\frac{1}{m}\right)^{c\rho k}$
- Overall failure probability is $\left(\frac{1}{m}\right)^{(c-1)\rho k}$
- So it holds with high probability.

Algorithm

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets

$\log n$ different guesses

$$\ell \in \{1,2,4,\dots,n\}$$

sample ℓ sets,

One pass, space $O(n)$, ℓ sets

Set Sampling: After picking ℓ sets uniformly at random, all elements with degree at least $\frac{\mathrm{m} \log n}{\ell}$ are covered w.h.p.

Algorithm

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For $1/\delta$ iterations
  - Use element sampling to cover $(1-\frac{1}{n^{\delta}})$-fraction of the uncovered elements.
  - Add the sets to Sol

$\log n$ different guesses

$$\ell \in \{1,2,4,\dots,n\}$$

sample ℓ sets,
One pass, space $O(n)$, ℓ sets

sample $(\rho \ell n^{\delta} \log m)$ elements,
In one pass, keep projection of sets on sampled elements
Space: $O\left(\rho \ell n^{\delta} \log m \cdot \frac{m \log n}{\ell}\right)$
$=O(\rho m n^{\delta} \log m \log n) = \tilde{O}(m n^{\delta})$
Compute a solution of $\rho\ell$ sets at the end

$\boldsymbol{\epsilon = 1/n^{\delta}}$

Element Sampling: Sampling $\Theta(\frac{\rho k \log m}{\epsilon})$ elements uniformly at random and finding a ρ-approximate cover for the sampled elements, will cover $(1-\epsilon)$ fraction of the original elements w.h.p.

Algorithm

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For $1/\delta$ iterations
  - Use element sampling to cover $(1-\frac{1}{n^{\delta}})$-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.

$\log n$ different guesses

$$\ell \in \{1,2,4,\dots,n\}$$

sample ℓ sets,

One pass, space $O(n)$, ℓ sets

sample $(\rho \ell n^{\delta} \log m)$ elements,

In one pass, keep projection of sets on sampled elements

Space: $O\left(\rho \ell n^{\delta} \log m \cdot \frac{m \log n}{\ell}\right)$

$=O\left(\rho m n^{\delta} \log m \log n\right) = \tilde{O}(m n^{\delta})$

Compute a solution of $\rho\ell$ sets at the end

One pass, space $O(n)$

Analysis

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For $1/\delta$ iterations
  - • Use element sampling to cover $(1-\frac{1}{n^{\delta}})$-fraction of the uncovered elements.
  - • Add the sets to Sol
  - • Update uncovered elements.

$\log n$ different guesses

$$\ell \in \{1,2,4,\dots,n\}$$

sample ℓ sets,
One pass, space $O(n)$, ℓ sets

sample $(\rho \ell n^{\delta} \log m)$ elements,
In one pass, keep projection of sets on sampled elements
Space: $O\left(\rho \ell n^{\delta} \log m \cdot \frac{m \log n}{\ell}\right)$
$=O(\rho m n^{\delta} \log m \log n) = \tilde{O}(m n^{\delta})$
Compute a solution of $\rho\ell$ sets at the end

One pass, space $O(n)$

Number of Passes: $2/\delta$

✓ $(\frac{1}{\delta})$ iterations, each 2 passes

✓ Set sampling pass can be merged with the first pass of the first iteration

Analysis

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For $1/\delta$ iterations
  - Use element sampling to cover $(1-\frac{1}{n^{\delta}})$-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.

$\log n$ different guesses

$$\ell \in \{1,2,4,\dots,n\}$$

sample ℓ sets,
One pass, space $\boldsymbol{O(n)}$, ℓ sets

sample $(\rho \ell n^{\delta} \log m)$ elements,
In one pass, keep projection of sets on sampled elements
Space: $O\left(\rho \ell n^{\delta} \log m \cdot \frac{m \log n}{\ell}\right)$
$=O(\rho m n^{\delta} \log m \log n) = \boldsymbol{\tilde{O}(mn^{\delta})}$
Compute a solution of $\rho\ell$ sets at the end

One pass, space $\boldsymbol{O(n)}$

Space Usage: $\boldsymbol{\tilde{O}(mn^{\delta})}$

- ✓ $\log n$ different guesses that run in parallel
- ✓ For each guess, the space usage is $\tilde{O}(mn^{\delta})$

Analysis

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For $1/\delta$ iterations
  - Use element sampling to cover $(1-\frac{1}{n^{\delta}})$-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.

$\log n$ different guesses

$$\ell \in \{1,2,4,\dots,n\}$$

sample ℓ sets,
One pass, space $O(n)$, ℓ **sets**

sample $(\rho\ell n^{\delta}\log m)$ elements,
In one pass, keep projection of sets on sampled elements
Space: $O\left(\rho\ell n^{\delta}\log m\cdot\frac{m\log n}{\ell}\right)$
$=O(\rho m n^{\delta}\log m\log n)=\tilde{O}(mn^{\delta})$
Compute a solution of $\boldsymbol{\rho\ell}$ **sets** at the end

One pass, space $O(n)$

Approximation Factor: $2\rho/\delta+2$

Take $k\le\ell\le 2k$

✓ $\ell\le 2k$ sets for set sampling

✓ $\rho\ell\le 2\rho k$ sets per pass

Analysis

Make a guess ℓ of the value of the optimal solution k

- ❑ Preprocessing: perform set sampling
- ❑ Sol ← sampled sets
- ❑ For $1/\delta$ iterations
  - Use element sampling to cover $(1-\frac{1}{n^{\delta}})$-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.

$\log n$ different guesses

$$\ell \in \{1,2,4,\dots,n\}$$

sample ℓ sets,
One pass, space $O(n)$, ℓ sets

sample $(\rho \ell n^{\delta} \log m)$ elements,
In one pass, keep projection of sets on sampled elements
Space: $O\left(\rho \ell n^{\delta} \log m \cdot \frac{m \log n}{\ell}\right)$
$=O(\rho m n^{\delta} \log m \log n) = \tilde{O}(m n^{\delta})$
Compute a solution of $\rho\ell$ sets at the end

One pass, space $O(n)$

Proof of correctness:

- ✓ For the right guess $k \le \ell < 2k$, every subsample of the elements can be covered by at most $k < \ell$ sets. Thus element sampling returns a cover of size at most $\rho\ell < 2\rho k$.
- ✓ The number of uncovered elements reduces by a factor of n^{δ}. So after $1/\delta$ iterations, all elements are covered.

What it gives

Algorithms	Approximation	Passes	Space	Type
Greedy Alg	$O(\log n)$	1	$O(mn)$	Deterministic
	$O(\log n)$	n	$O(n)$	Deterministic
[GS'09]	$O(\log n)$	$O(\log n)$	$O(n \log n)$	Deterministic
[ER'14]	$O(\sqrt{n})$	1	$\tilde{O}(n)$	Deterministic
[CW'16]	$O(n^{\delta}/\delta)$	$1/\delta - 1$	$\tilde{O}(n)$	Deterministic
[HIMV'16]	$\mathbf{O(\rho/\delta)}$	$\mathbf{O(1/\delta)}$	$\mathbf{\tilde{O}(mn^{\delta})}$	Randomized

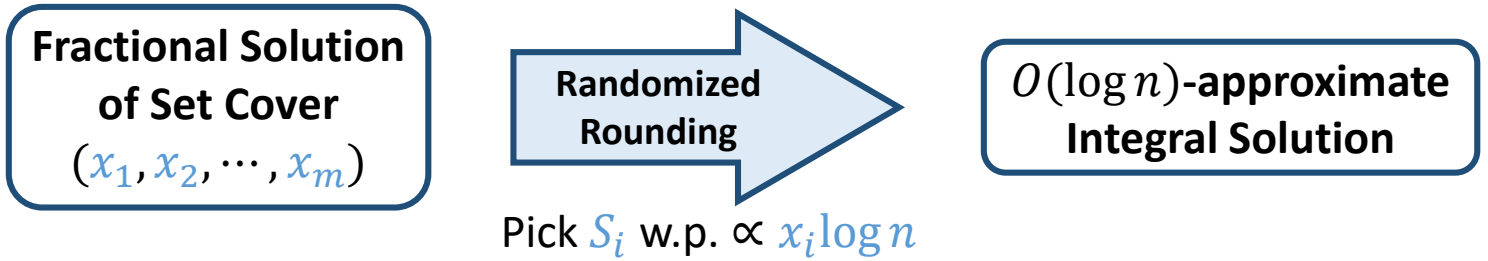
ρ = approximation guarantee
for offline **Set Cover**

n = number of *elements*
m = *number of sets.*

Fractional Set Cover using MWU

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]

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 packing/covering 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.
- $T = \mathrm{O}(\phi \log n \,/\epsilon^2)$

MWU to solve LP

$$\begin{pmatrix} a_{1,1}, \dots, a_{1,m} \\ a_{2,1}, \dots, a_{2,m} \\ \dots \\ a_{n,1}, \dots, a_{n,m} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_m \end{pmatrix} \geq \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

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

Min $c^T x$

$Ax \geq b$

$x \geq 0$

Covering LP:
All $a_{j,i}$ and b_j and c_i are non-negative

MWU to solve LP

Algorithm:

- Instead of solving for all the constraints, solve for a weighted average constraint.

$$\begin{matrix} w_1 \\ w_2 \\ \dots \\ w_n \end{matrix} \begin{pmatrix} a_{1,1}, \dots, a_{1,m} \\ a_{2,1}, \dots, a_{2,m} \\ \dots \\ a_{n,1}, \dots, a_{n,m} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_m \end{pmatrix} \geq \begin{matrix} w_1 \\ w_2 \\ \dots \\ w_n \end{matrix} \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

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.

$$\begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}^T \begin{pmatrix} a_{1,1}, \dots, a_{1,m} \\ a_{2,1}, \dots, a_{2,m} \\ \dots \\ a_{n,1}, \dots, a_{n,m} \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_m \end{bmatrix} \geq \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}^T \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

$$\sum_{i=1}^{m} x_i \left(\sum_{j=1}^{n} w_j a_{j,i} \right) \geq \sum_{j=1}^{n} w_j b_j$$

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.
- $T = O(\phi \log n / \epsilon^2)$

$w^1 \leftarrow (1, \cdots, 1)$ ▷ 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 $c^T x$

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

$x \geq 0$

MWU Update Rule:

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

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

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

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)

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

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

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

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

Solve the *weighted* average constraint approximately!

Multiplicative Weight Update (Set Cover)

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

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

$$\sum_{e \in \mathcal{U}} w_e \sum_{e \in S} x_S \geq \sum_{e \in \mathcal{U}} w_e$$
$$x_S \geq 0 \qquad \forall S \in \mathcal{F}$$

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

Solve the *weighted* average constraint approximately!

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

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

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

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

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

T times

Assign weight w_e to each element e (initially one)

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

Update the prob vector

$$\boldsymbol{p_e^{t+1} := p_e^t(1 - O(\varepsilon) \times (\sum x_S^t - 1))}$$

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

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

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

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

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

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

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

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

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

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

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.
- ✓ 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

- ➢ 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 $\check{\mathcal{F}}$

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 $\mathcal{F}$.

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

$$\check{\mathcal{F}} = \{$$

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

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

$$\{\mathbf{1},\mathbf{2},\mathbf{3}\},\{3,4,5\}$$

$$\}$$

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., $x_S = \begin{cases} 1 \\ 0 \end{cases}$ If S is one of the **k heaviest** set, Otherwise.
  - 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;
$|\cup_{S \in \mathcal{C}} S|$

Complexity:

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

Fractional Max *k*-Cover

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

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

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

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

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

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

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 $x^{\text{cmn}}=\left\langle \frac{\varepsilon k}{m}, \dots, \frac{\varepsilon k}{m} \right\rangle$
  - 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
  - $\boldsymbol{p_e^{t+1} := p_e^t\left(1 - O(\varepsilon) \times (y_e{}' - 1)\right)}$

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

Lemma: 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 $O\left(\frac{\log n}{\varepsilon^2}\right)$

Performance

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

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

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

$$\sum_{S\in\mathcal{F}} x_S p_S \ge 1-\varepsilon$$
$$x_S \ge 0 \qquad \forall S \in \mathcal{F}$$
$$-1 \le \sum_{S:e\in S} x_S - 1 \le 1 \quad \forall e \in \mathcal{U}$$

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.

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

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

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:

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

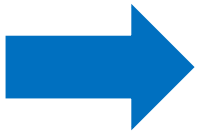
Setting $\ell = O(\frac{\log n}{\varepsilon^2 d})$ rounds, p_e increases at most by $n^{O(\frac{1}{\varepsilon d})}$

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

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

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

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

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

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**	$O(1/\delta)$-**pass**	$\tilde{O}(mn^{O(\delta/\varepsilon)})$-**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.
- 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
- 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
$O(k \log n / \epsilon^2)$ passes
$\tilde{O}(n)$ space

Performance

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

Performance

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