Set Cover in Sub-linear Time

Piotr Indyk
MIT

Sepideh Mahabadi
Columbia University

Ronitt Rubinfeld
MIT/TAU

Ali Vakilian
MIT

Anak Yodpinyanee
MIT
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
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 $S_1, \ldots, S_m$, 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
- Can’t do better unless $P=NP$ [LY91][RS97][Fei98][AMS06][DS14]
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
  - Can’t do better unless $P=NP$ \cite{LY91,RS97,Fei98,AMS06,DS14}

“Is it possible to solve minimum set cover in sub-linear time?”
Sub-linear Time Set Cover

Data Access Model?
Sub-linear Time Set Cover

Data Access Model [NO’08,YYI’12]

\[
\text{EltOf}(S, i): \text{ith element in } S \\
\text{SetOf}(e, j): \text{jth set containing } e
\]
Sub-linear Time Set Cover

**Data Access Model** [NO’08, YYI’12]

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs

$\text{EltOf}(S, i)$: $i$th element in $S$

$\text{SetOf}(e, j)$: $j$th set containing $e$
Sub-linear Time Set Cover

**Data Access Model** [NO’08, YYI’12]
- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in \( mn \)

\[
\text{EltOf}(S, i) \colon \text{ith element in } S
\]
\[
\text{SetOf}(e, j) \colon \text{jth set containing } e
\]
Sub-linear Time Set Cover

**Data Access Model** [NO’08, YYI’12]

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in $mn$

**Prior Results**

EltOf($S, i$): $i$th element in $S$
SetOf($e, j$): $j$th set containing $e$
Sub-linear Time Set Cover

**Data Access Model** [NO’08, YYI’12]

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in $mn$

**Prior Results**

- [Nguyen, Onak’08][Yoshida, Yamamoto, Ito’12]
  - Constant queries, if degree is constant

$\text{EltOf}(S, i)$: $i$th element in $S$

$\text{SetOf}(e, j)$: $j$th set containing $e$
Sub-linear Time Set Cover

**Data Access Model** [NO’08, YYI’12]

- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in $mn$

**Prior Results**

- [Nguyen, Onak’08][Yoshida, Yamamoto, Ito’12]:
  - Constant queries, if degree is constant
- [Koufogiannakis, Young’14][Grigoriadis, Kachiyan’95]:
  - Find $(1 + \epsilon)$-approximate *fractional solution*, then perform *randomized rounding* to achieve $O(\log n)$-approximation

**Definitions**

- $\text{EltOf}(S, i)$: $i$th element in $S$
- $\text{SetOf}(e, j)$: $j$th set containing $e$
Sub-linear Time Set Cover

**Data Access Model** [NO’08,YYI’12]
- No assumption on the order
- Incidence list in (sub-linear) algorithms for graphs
- Sublinear in $mn$

**Prior Results**
- [Nguyen, Onak’08][Yoshida, Yamamoto, Ito’12]
  - Constant queries, if degree is constant
- [Koufogiannakis, Young’14][Grigoriadis, Kachiyan’95]:
  - Find $(1 + \epsilon)$-approximate **fractional solution**, then perform **randomized rounding** to achieve $O(\log n)$-approximation
  - $O(mk^2 + nk^2)$ (can be improved to $O(m + nk)$)

$n =$ number of elements $\quad m =$ number of sets $\quad k =$ size of the optimal solution
## Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Constraints</th>
<th>Query Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set Cover</strong></td>
<td>$\alpha \rho + 1$</td>
<td>$\alpha \geq 2$</td>
<td>$\tilde{O} \left( m \left( \frac{n}{k} \right)^{\frac{1}{\alpha-1}} + nk \right)$</td>
</tr>
<tr>
<td>$\rho + 1$</td>
<td>$\rho + 1$</td>
<td>$-$</td>
<td>$\tilde{O} \left( \frac{mn}{k} \right)$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$k \leq \left( \frac{n}{\log m} \right)^{\frac{1}{4\alpha+1}}$</td>
<td>$\tilde{O} \left( m \left( \frac{n}{k} \right)^{\frac{1}{2\alpha}} \right)$</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\alpha \leq 1.01$</td>
<td>$k = O(n/\log m)$</td>
<td>$\tilde{O} \left( \frac{mn}{k} \right)$</td>
</tr>
</tbody>
</table>

| **Cover Verification**       | $-$           | $k \leq n/2$ | $\tilde{\Omega}(nk)$ |

$\rho$ = approximation factor for offline **Set Cover**

$n$ = number of *elements*  \quad $m$ = number of *sets*  \quad $k$ = Size of the optimal Solution
## Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Constraints</th>
<th>Query Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set Cover</strong></td>
<td>$\alpha \rho + 1$</td>
<td>$\alpha \geq 2$</td>
<td>$\tilde{O}\left(m \left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$</td>
</tr>
<tr>
<td></td>
<td>$\rho + 1$</td>
<td>$-$</td>
<td>$\tilde{O}\left(\frac{mn}{k}\right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$</td>
<td>$\tilde{\Omega}\left(m \left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\alpha \leq 1.01$</td>
<td>$\tilde{\Omega}\left(\frac{mn}{k}\right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$k = O(n/\log m)$</td>
<td>$\tilde{\Omega}(nk)$</td>
</tr>
<tr>
<td><strong>Cover Verification</strong></td>
<td>$-$</td>
<td>$k \leq n/2$</td>
<td>$\tilde{\Omega}(nk)$</td>
</tr>
</tbody>
</table>

$\rho = \text{approximation factor for offline Set Cover}$  
$n = \text{number of elements} \quad m = \text{number of sets} \quad k = \text{Size of the optimal Solution}$
## Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Constraints</th>
<th>Query Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Cover</td>
<td>$\alpha \rho + 1$</td>
<td>$\alpha \geq 2$</td>
<td>$\tilde{O}\left(m \left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$</td>
</tr>
<tr>
<td></td>
<td>$\rho + 1$</td>
<td>$-$</td>
<td>$\tilde{O}\left(\frac{mn}{k}\right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$</td>
<td>$\tilde{\Omega}\left(m \left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\alpha \leq 1.01$</td>
<td>$\tilde{\Omega}\left(\frac{mn}{k}\right)$</td>
</tr>
<tr>
<td></td>
<td>$k = O(n/\log m)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cover Verification</td>
<td>$-$</td>
<td>$k \leq n/2$</td>
<td>$\tilde{\Omega}(nk)$</td>
</tr>
</tbody>
</table>

**Cover Verification**: given a set system, verify whether a given sub-collection of sets covers the universe.

$\rho$ = approximation factor for offline **Set Cover**

$n = \text{number of elements} \quad m = \text{number of sets} \quad k = \text{Size of the optimal Solution}$
## Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Constraints</th>
<th>Query Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set Cover</strong></td>
<td>$\alpha \rho + 1$</td>
<td>$\alpha \geq 2$</td>
<td>$\tilde{O}\left(m \left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$</td>
</tr>
<tr>
<td></td>
<td>$\rho + 1$</td>
<td>$-$</td>
<td>$\tilde{O}\left(\frac{mn}{k}\right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$</td>
<td>$\tilde{\Omega}\left(m \left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\alpha \leq 1.01$</td>
<td>$\tilde{\Omega}\left(\frac{mn}{k}\right)$</td>
</tr>
<tr>
<td></td>
<td>$k = O(n/\log m)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cover Verification</strong></td>
<td>$-$</td>
<td>$k \leq n/2$</td>
<td>$\tilde{\Omega}(nk)$</td>
</tr>
</tbody>
</table>

**Cover Verification**: given a set system, verify whether a given sub-collection of sets covers the universe.

$\rho = $ approximation factor for offline **Set Cover**

$n =$ number of elements \hspace{1cm} $m =$ number of sets \hspace{1cm} $k =$ Size of the optimal Solution
## Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Constraints</th>
<th>Query Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set Cover</strong></td>
<td>$\alpha \rho + 1$</td>
<td>$\alpha \geq 2$</td>
<td>$\tilde{O}\left( m \left( \frac{n}{k} \right)^{\frac{1}{\alpha-1}} + nk \right)$</td>
</tr>
<tr>
<td></td>
<td>$\rho + 1$</td>
<td>$-$</td>
<td>$\tilde{O}\left( \frac{mn}{k} \right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$k \leq \left( \frac{n}{\log m} \right)^{\frac{1}{4\alpha+1}}$</td>
<td>$\tilde{\Omega}\left( m \left( \frac{n}{k} \right)^{\frac{1}{2\alpha}} \right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\alpha \leq 1.01$</td>
<td>$\tilde{\Omega}\left( \frac{mn}{k} \right)$</td>
</tr>
<tr>
<td></td>
<td>$k = O(n/\log m)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cover Verification</strong></td>
<td>$-$</td>
<td>$k \leq n/2$</td>
<td>$\tilde{\Omega}(nk)$</td>
</tr>
</tbody>
</table>

**Cover Verification**: given a set system, verify whether a given sub-collection of sets covers the universe.

**$\rho$** = approximation factor for offline **Set Cover**

**$n$** = number of **elements** \hspace{1em} **$m$** = number of **sets** \hspace{1em} **$k$** = Size of the optimal Solution
## Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Constraints</th>
<th>Query Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set Cover</strong></td>
<td>$\alpha \rho + 1$</td>
<td>$\alpha \geq 2$</td>
<td>$\tilde{\Omega} \left( m \left( \frac{n}{k} \frac{1}{\alpha-1} \right) + nk \right)$</td>
</tr>
<tr>
<td></td>
<td>$\rho + 1$</td>
<td>$-$</td>
<td>$\tilde{\Omega} \left( \frac{mn}{k} \right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$k \leq \left( \frac{n}{\log m} \right)^{\frac{1}{4\alpha+1}}$</td>
<td>$\tilde{\Omega} \left( m \left( \frac{n}{k} \frac{1}{2\alpha} \right) \right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\alpha \leq 1.01$</td>
<td>$\tilde{\Omega} \left( \frac{mn}{k} \right)$</td>
</tr>
<tr>
<td></td>
<td>$k = O(n/\log m)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cover Verification</strong></td>
<td>$-$</td>
<td>$k \leq n/2$</td>
<td>$\tilde{\Omega}(nk)$</td>
</tr>
</tbody>
</table>

**Cover Verification**: given a set system, verify whether a given sub-collection of sets covers the universe.

$\rho$ = approximation factor for offline Set Cover

$n = \text{number of elements}$, $m = \text{number of sets}$, $k = \text{Size of the optimal Solution}$
## Cover Verification

: given a set system, verify whether a given sub-collection of sets covers the universe.

\[ \rho = \text{approximation factor for offline Set Cover} \]

\[ n = \text{number of elements} \quad m = \text{number of sets} \quad k = \text{Size of the optimal Solution} \]
Part one: upper bound

**Theorem:** There exists an algorithm that with high probability finds an $O(\rho \alpha)$-approximate cover which uses $\tilde{O}(mn^{1/\alpha} + nk)$ number of queries.
Theorem: There exists an algorithm that with high probability finds an $O(\rho \alpha)$-approximate cover which uses $\tilde{O}(mn^{1/\alpha} + nk)$ number of queries.

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

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

- We only need to worry about low degree elements.
Component I: set sampling

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

- We only need to worry about low degree elements.

How we use the lemma: set $\ell = O(k)$
Component I: set sampling

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

- We only need to worry about low degree elements.

\[ \ell = 2 \]
Component I: set sampling

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

- We only need to worry about low degree elements.

$\ell = 2$

Degrees: 2 3 2 1 1 3 2 1 3 2
**Set Sampling**: After picking $\ell$ sets uniformly at random, all elements with degree at least $\frac{m \log n}{\ell}$ are covered w.h.p.

- We only need to worry about low degree elements.

$\ell = 2$

Degrees: 2 3 2 1 1 3 2 1 3 2
Set Sampling: After picking $\ell$ sets uniformly at random, all elements with degree at least $\frac{m \log n}{\ell}$ 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.
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 $\Theta\left(\frac{\rho k \log m}{\delta}\right)$ elements uniformly at random and finding a $\rho$-approximate cover for the sampled elements, will cover $(1 - \delta)$ fraction of the original elements w.h.p.
Algorithm

Make a guess $\ell$ of the value of the optimal solution $k$.
Algorithm

Make a guess $\ell$ of the value of the optimal solution $k$

$log n$ different guesses
$\ell \in \{1,2,4,\ldots,n\}$
Algorithm

Make a guess $\ell$ of the value of the optimal solution $k$

- Preprocessing: perform set sampling
- $\text{Sol} \leftarrow$ sampled sets

$log n$ different guesses $\ell \in \{1, 2, 4, \ldots, n\}$
Algorithm

Make a **guess** \( \ell \) of the value of the optimal solution \( k \)

- Preprocessing: perform **set sampling**
- Sol \( \leftarrow \) sampled sets

\[ \log n \] different guesses
\[ \ell \in \{1,2,4,\ldots,n\} \]

Sample \( \ell \) sets,
**number of queries**: \( n\ell \)

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

Make a **guess** $\ell$ of the value of the optimal solution $k$

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

$log n$ different guesses $\ell \in \{1, 2, 4, \ldots, n\}$

Sample $\ell$ sets, number of queries: $n\ell$
Algorithm

Make a guess $\ell$ of the value of the optimal solution $k$
- **Preprocessing**: perform set sampling
- Sol $\leftarrow$ sampled sets
- For $\alpha$ iterations
  - Use **element sampling** to cover $(1 - \frac{1}{n^{1/\alpha}})$-fraction of the uncovered elements.
  - Add the sets to Sol

log $n$ different guesses
$\ell \in \{1, 2, 4, \ldots, n\}$

sample $\ell$ sets, number of queries: $n\ell$

$\delta = \frac{1}{n^{1/\alpha}}$

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

Make a **guess** $\ell$ of the value of the optimal solution $k$

- **Preprocessing**: perform set sampling
- $\text{Sol} \leftarrow$ sampled sets
- For $\alpha$ iterations
  - Use **element sampling** to cover $\left(1 - \frac{1}{n^{1/\alpha}}\right)$ fraction of the uncovered elements.
  - Add the sets to $\text{Sol}$

$\log n$ different guesses $\ell \in \{1, 2, 4, \ldots, n\}$

Sample $\ell$ sets, number of queries: $n\ell$

Sample $(\rho \ell n^{1/\alpha} \log m)$ elements,
number of queries:

$O\left(\rho \ell n^{1/\alpha} \log m \frac{m \log n}{\ell}\right) = O(\rho mn^{1/\alpha} \log m \log n)$

$\delta = 1/n^{1/\alpha}$

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

Make a guess $\ell$ of the value of the optimal solution $k$

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

$log n$ different guesses
$\ell \in \{1, 2, 4, \ldots, n\}$

Sample $\ell$ sets,
number of queries: $n\ell$

Sample $(\rho \ell n^{1/\alpha} \log m)$ elements,
number of queries:
$O \left( \rho \ell n^{1/\alpha} \log m \frac{m \log n}{\ell} \right)$
$= O(\rho mn^{1/\alpha} \log m \log n)$
Algorithm

Make a **guess** $\ell$ of the value of the optimal solution $k$

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

$log n$ different guesses $\ell \in \{1, 2, 4, \ldots, n\}$

Sample $\ell$ sets, number of queries: $n\ell$

Sample $(\rho \ell n^{1/\alpha} \log m)$ elements, number of queries:

$$O \left( \rho \ell n^{1/\alpha} \log m \frac{m \log n}{\ell} \right)$$

$$= O(\rho mn^{1/\alpha} \log m \log n)$$

Number of queries: $\rho n\ell$
Algorithm

Make a **guess** \( \ell \) of the value of the optimal solution \( k \)
- **Preprocessing:** perform set sampling
- Sol ← sampled sets
- For \( \alpha \) iterations
  - Use **element sampling** to cover \( (1 - \frac{1}{n^{1/\alpha}}) \)-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.
- If all elements are covered, report Sol

\( \log n \) different guesses
\( \ell \in \{1,2,4, \ldots, n\} \)

Sample \( \ell \) sets,
number of queries: \( n\ell \)

Sample \( (\rho \ell n^{1/\alpha} \log m) \)
elements,
number of queries:
\( O \left( \rho \ell n^{1/\alpha} \log m \frac{m \log n}{\ell} \right) \)
= \( O(\rho \ell n^{1/\alpha} \log m \log n) \)

Number of queries: \( \rho n\ell \)
Algorithm

Make a guess $\ell$ of the value of the optimal solution $k$

- **Preprocessing**: perform set sampling
- Sol $\leftarrow$ sampled sets
- For $\alpha$ iterations
  - Use element sampling to cover $(1 - \frac{1}{n^{1/\alpha}})$-fraction of the uncovered elements.
  - Add the sets to Sol
  - Update uncovered elements.
- If all elements are covered, report Sol

**Theorem**: There exists an algorithm that with high probability finds an $O(\rho \alpha)$-approximate cover which uses $\tilde{O}(mn^{1/\alpha} + nk)$ number of queries.
## Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Constraints</th>
<th>Query Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set Cover</strong></td>
<td>$\alpha \rho + 1$</td>
<td>$\alpha \geq 2$</td>
<td>$\tilde{O} \left( m \left( \frac{n}{k} \right)^{\frac{1}{\alpha-1}} + nk \right)$</td>
</tr>
<tr>
<td></td>
<td>$\rho + 1$</td>
<td>$-$</td>
<td>$\tilde{O} \left( \frac{mn}{k} \right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$k \leq \left( \frac{n}{\log m} \right)^{\frac{1}{4\alpha+1}}$</td>
<td>$\tilde{\Omega} \left( m \left( \frac{n}{k} \right)^{\frac{1}{2\alpha}} \right)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\alpha \leq 1.01$</td>
<td>$\tilde{\Omega} \left( \frac{mn}{k} \right)$</td>
</tr>
<tr>
<td></td>
<td>$k = O(n/\log m)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cover Verification</strong></td>
<td>$-$</td>
<td>$k \leq n/2$</td>
<td>$\tilde{\Omega}(nk)$</td>
</tr>
</tbody>
</table>

**Cover Verification**: given a set system, verify whether a given sub-collection of sets covers the universe.

$\rho$ = approximation factor for offline **Set Cover**

$n$ = number of *elements*  \hspace{1cm}  $m$ = number of *sets*  \hspace{1cm}  $k$ = Size of the optimal Solution
Part two: lower bound

**Theorem:** Any randomized algorithm that with probability at least $2/3$ distinguishes whether the minimum Set Cover size is 2 or at least 3 requires $\tilde{\Omega}(mn)$ number of queries.
High Level Approach

1. Construct a median instance $I^*$
   • Minimum Set Cover Size is 3
1. **Construct a median instance** $I^*$
   - Minimum Set Cover Size is 3
2. **Randomized Procedure** on $I^*$ **to get a modified instance** $I$
   - Minimum Set Cover Size is 2
   - $I^*$ and $I$ only differ in a few positions
   - The differences are distributed almost uniformly at random
High Level Approach

1. **Construct a median instance** $I^*$
   - Minimum Set Cover Size is 3

2. **Randomized Procedure** on $I^*$ to get a modified instance $I$
   - Minimum Set Cover Size is 2
   - $I^*$ and $I$ only differ in a few positions
   - The differences are distributed almost uniformly at random

3. Any algorithm that can detect these two cases requires to query at least $\tilde{\Omega}(mn)$ queries.
The Median Instance

**Construction:** is randomized. For every \( S, e \) the set \( S \) contains \( e \) with probability \( 1 - p_0 \) where \( p_0 = \sqrt{\frac{9 \log m}{n}} \).
The Median Instance

Construction: is randomized. For every $S, e$ the set $S$ contains $e$ with probability $1 - p_0$ where $p_0 = \sqrt{\frac{9 \log m}{n}}$

Properties: by Chernoff, most of such instances have the following properties:

1. No 2 sets cover all the elements
2. For any two sets the number of uncovered elements is $O(\log m)$
3. The intersection is at least $\Omega(n)$
4. For each element, the number of sets not covering it is at most $6m \sqrt{\frac{\log m}{n}}$
5. For any pair of elements the number of sets containing only the first element is at least $\frac{m \sqrt{9 \log m}}{4\sqrt{n}}$
6. For any three sets, the number of elements in the first two but not in the third one is at least $6\sqrt{n \log m}$
The Median Instance

Construction: is randomized. For every \( S, e \) the set \( S \) contains \( e \) with probability \( 1 - p_0 \) where \( p_0 = \sqrt{\frac{9 \log m}{n}} \).

Properties: by Chernoff, most of such instances have the following properties:

1. No 2 sets cover all the elements
2. For any two sets the number of uncovered elements is \( O(\log m) \)
3. The intersection is at least \( \Omega(n) \)
4. For each element, the number of sets not covering it is at most \( 6m \sqrt{\frac{\log m}{n}} \)
5. For any pair of elements the number of sets containing only the first element is at least \( \frac{m \sqrt{9 \log m}}{4 \sqrt{n}} \)
6. For any three sets, the number of elements in the first two but not in the third one is at least \( 6 \sqrt{n \log m} \)
The Median Instance

**Construction:** is randomized. For every $S, e$ the set $S$ contains $e$ with probability $1 - p_0$ where $p_0 = \sqrt{\frac{9 \log m}{n}}$

**Properties:** by Chernoff, most of such instances have the following properties:

1. No 2 sets cover all the elements
2. For any two sets the number of uncovered elements is $O(\log m)$
3. The intersection is at least $\Omega(n)$
4. For each element, the number of sets not covering it is at most $6m \sqrt{\frac{\log m}{n}}$
5. For any pair of elements the number of sets containing only the first element is at least $\frac{m \sqrt{9 \log m}}{4\sqrt{n}}$
6. For any three sets, the number of elements in the first two but not in the third one is at least $6 \sqrt{n \log m}$

Take one such instance $I^*$ with the above properties
# The Median Instance

<table>
<thead>
<tr>
<th>Sets</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e \in S$</td>
<td>Grey</td>
</tr>
<tr>
<td>$e \notin S$</td>
<td>White</td>
</tr>
</tbody>
</table>
Generating a Modified Instance

Pick two random sets $S_1$ and $S_2$ and turn them into a set cover. How?

$U = \{e_1, e_2, e_3, e_4\}$

$S_1 = \{e_2, e_3\}$

$S_2 = \{e_2, e_4\}$
Generating a Modified Instance

Pick two random sets $S_1$ and $S_2$ and turn them into a set cover. How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add $e_1$ to $S_2$

$U = \{e_1, e_2, e_3, e_4\}$

$S_1 = \{e_2, e_3\}$

$S_2 = \{e_2, e_4\}$  $\leftarrow e_1$
Generating a Modified Instance

Pick two random sets $S_1$ and $S_2$ and turn them into a set cover. How?

• For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  • Add $e_1$ to $S_2$
  • Remove an element $e_2 \in S_2 \cap S_1$ from $S_2$

$U = \{e_1, e_2, e_3, e_4\}$

$S_1 = \{e_2, e_3\}$

$S_2 = \{e_2, e_4\}$
Generating a Modified Instance

Pick two random sets $S_1$ and $S_2$ and turn them into a set cover. How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add $e_1$ to $S_2$
  - Remove an element $e_2 \in S_2 \cap S_1$ from $S_2$
  - Pick a random set $S_3$ that contains $e_1$ but not $e_2$

$U = \{e_1, e_2, e_3, e_4\}$

$S_1 = \{e_2, e_3\}$

$S_2 = \{e_2, e_4\}$

$S_3 = \{e_4, e_1\}$
Generating a Modified Instance

Pick two random sets $S_1$ and $S_2$ and turn them into a set cover. How?
- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add $e_1$ to $S_2$
  - Remove an element $e_2 \in S_2 \cap S_1$ from $S_2$
  - Pick a random set $S_3$ that contains $e_1$ but not $e_2$
  - $S_2$ and $S_3$ swap $e_1$ and $e_2$

$U = \{e_1, e_2, e_3, e_4\}$

$S_1 = \{e_2, e_3\}$

$S_2 = \{e_2, e_4\}$

$S_3 = \{e_4, e_1\}$
Generating a Modified Instance

Pick two random sets $S_1$ and $S_2$ and turn them into a set cover. How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add $e_1$ to $S_2$
  - Remove an element $e_2 \in S_2 \cap S_1$ from $S_2$
  - Pick a random set $S_3$ that contains $e_1$ but not $e_2$
  - $S_2$ and $S_3$ swap $e_1$ and $e_2$

$U = \{e_1, e_2, e_3, e_4\}$

Modified instance:

- $S_1 = \{e_2, e_3\}$
- $S_2 = \{e_1, e_4\}$
- $S_3 = \{e_4, e_2\}$
Generating a Modified Instance

Pick two random sets $S_1$ and $S_2$ and turn them into a set cover. How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add $e_1$ to $S_2$
  - Remove an element $e_2 \in S_2 \cap S_1$ from $S_2$
  - Pick a random set $S_3$ that contains $e_1$ but not $e_2$
  - $S_2$ and $S_3$ swap $e_1$ and $e_2$

$U = \{e_1, e_2, e_3, e_4\}$

Modified instance:

- $S_1 = \{e_2, e_3\}$
- $S_2 = \{e_1, e_4\}$
- $S_3 = \{e_4, e_2\}$

Only four positions changes in the query access model.
Generating a Modified Instance

Pick two random sets $S_1$ and $S_2$ and turn them into a set cover. How?

- For each uncovered element $e_1 \in U \setminus (S_1 \cup S_2)$,
  - Add $e_1$ to $S_2$
  - Remove an element $e_2 \in S_2 \cap S_1$ from $S_2$
  - Pick a random set $S_3$ that contains $e_1$ but not $e_2$
  - $S_2$ and $S_3$ swap $e_1$ and $e_2$

$U = \{e_1, e_2, e_3, e_4\}$

Modified instance

$S_1 = \{e_2, e_3\}$
$S_2 = \{e_1, e_4\}$
$S_3 = \{e_4, e_2\}$

Only four positions changes in the query access model.
The Randomized Procedure

- Median Instance
- **Pick two Sets**
  Uniformly at Random

\[ S_1 \]

\[ S_2 \]
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- **Also find the elements that are covered by both**
• Median Instance
• Pick two Sets Uniformly at Random
• Find the elements that are not covered
• Also find the elements that are covered by both
• Assign one element in the intersection to each uncovered element
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- In iteration:
  - Find a candidate set
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element

**In iteration:**
- Find a candidate set
- **swap**
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element

**In iteration:**
- Find a candidate set
- swap
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element

**In iteration:**
- Find a candidate set
- swap
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- In iteration:
  - Find a candidate set
  - swap

\[ S_1 \]

\[ S_2 \]
Median Instance
Pick two Sets Uniformly at Random
Find the elements that are not covered
Also find the elements that are covered by both
Assign one element in the intersection to each uncovered element

In iteration:
  • Find a candidate set
  • swap
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element

**In iteration:**
- Find a candidate set
- **swap**
The Randomized Procedure

• Median Instance
• Pick two Sets Uniformly at Random
• Find the elements that are not covered
• Also find the elements that are covered by both
• Assign one element in the intersection to each uncovered element
• In iteration:
  • Find a candidate set
  • swap
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- In iteration:
  - Find a candidate set
  - swap

\[
\begin{array}{cccccccccccccccccc}
\text{S1} & \text{S2} \\
\end{array}
\]
The Randomized Procedure

- Median Instance
- Pick two Sets Uniformly at Random
- Find the elements that are not covered
- Also find the elements that are covered by both
- Assign one element in the intersection to each uncovered element
- In iteration:
  - Find a candidate set
  - swap

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- By Property 2 of median instance:
  - the total number of uncovered elements is $O(\log m)$
- Thus in total only $O(\log m)$ positions have changed.
Lemma: For any element $e$ and any set $S$, the probability that pair participate in a swap is almost uniform, i.e., $O\left(\frac{\log m}{mn}\right)$.

- Using other properties of the median instances

Input:
- W.p. $\frac{1}{2}$ the input is the median instance $I^*$
- W.p. $\frac{1}{2}$ the input is a randomly generated modified instance $I$
Lemma: For any element \(e\) and any set \(S\), the probability that pair participate in a swap is almost uniform, i.e., \(O\left(\frac{\log m}{mn}\right)\).

- Using other properties of the median instances

Input:
- W.p. \(\frac{1}{2}\) the input is the median instance \(I^*\)
- W.p. \(\frac{1}{2}\) the input is a randomly generated modified instance \(I\)

Theorem: Any randomized algorithm that with probability at least \(\frac{2}{3}\) distinguishes whether the minimum Set Cover size is 2 or at least 3 requires \(\tilde{\Omega}(mn)\) number of queries.
Open Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Query Complexity</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Cover</td>
<td>$\alpha\rho + 1$</td>
<td>$\tilde{O}\left(m \left(\frac{n}{k}\right)^{\frac{1}{\alpha-1}} + nk\right)$</td>
<td>$\alpha \geq 2$</td>
</tr>
<tr>
<td></td>
<td>$\rho + 1$</td>
<td>$\tilde{O}\left(\frac{mn}{k}\right)$</td>
<td>$-$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\tilde{\Omega}\left(m \left(\frac{n}{k}\right)^{\frac{1}{2\alpha}}\right)$</td>
<td>$k \leq \left(\frac{n}{\log m}\right)^{\frac{1}{4\alpha+1}}$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\tilde{\Omega}\left(\frac{mn}{k}\right)$</td>
<td>$\alpha \leq 1.01$ $k = O(n/\log m)$</td>
</tr>
<tr>
<td>Cover Verification</td>
<td>$-$</td>
<td>$\tilde{\Omega}(nk)$</td>
<td>$k \leq n/2$</td>
</tr>
</tbody>
</table>

- Prove a lower bound of $\Omega(nk)$ for the set cover problem as well
## Open Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Query Complexity</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set Cover</strong></td>
<td>$\alpha \rho + 1$</td>
<td>$\tilde{O} \left( m \left( \frac{n}{k} \right)^{\frac{1}{\alpha - 1}} + nk \right)$</td>
<td>$\alpha \geq 2$</td>
</tr>
<tr>
<td></td>
<td>$\rho + 1$</td>
<td>$\tilde{O} \left( \frac{mn}{k} \right)$</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\tilde{\Omega} \left( m \left( \frac{n}{k} \right)^{\frac{1}{2\alpha}} \right)$</td>
<td>$k \leq \left( \frac{n}{\log m} \right)^{\frac{1}{4\alpha + 1}}$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>$\tilde{\Omega} \left( \frac{mn}{k} \right)$</td>
<td>$\alpha \leq 1.01 \quad k = O(n/\log m)$</td>
</tr>
<tr>
<td><strong>Cover Verification</strong></td>
<td>–</td>
<td>$\tilde{\Omega}(nk)$</td>
<td>$k \leq n/2$</td>
</tr>
</tbody>
</table>

- Prove a lower bound of $\Omega(nk)$ for the set cover problem as well
- Similar results for the weighted set cover?
Open Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approximation</th>
<th>Query Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Cover</td>
<td>( \alpha \rho + 1 )</td>
<td>( \tilde{O} \left( m \frac{(n)^{\frac{1}{\alpha-1}}}{k} + nk \right) )</td>
</tr>
<tr>
<td>Set Cover</td>
<td>( \rho + 1 )</td>
<td>( \tilde{O} \left( \frac{mn}{k} \right) )</td>
</tr>
<tr>
<td>Set Cover</td>
<td>( \alpha )</td>
<td>( \tilde{\Omega} \left( m \frac{(n)^{\frac{1}{2\alpha}}}{k} \right) )</td>
</tr>
<tr>
<td>Set Cover</td>
<td>( \alpha )</td>
<td>( \tilde{\Omega} \left( \frac{mn}{k} \right) )</td>
</tr>
</tbody>
</table>

Cover Verification

- \( \tilde{\Omega}(nk) \)
  - \( k \leq n/2 \)

- Prove a lower bound of \( \Omega(nk) \) for the set cover problem as well
- Similar results for the weighted set cover?